Feedback compensation techniques to improve input disturbance response in the Cuk converter

Andrew Chuinard
Tim Chairet

June 10, 2009

Abstract

Constant output voltage is an important feature of a DC voltage regulator. This paper describes the derivation of the equations that make up the loop gain for a Cuk converter, and how to design a compensation scheme to minimize output voltage deviation in response to change in the input voltage of the system. Through simulation it is confirmed that a PID compensation method improves the output response of the system.

Introduction

In DC-DC voltage regulators, it is important to supply a constant output voltage, regardless of disturbances on the input voltage. The goal of this paper is to describe a feedback compensation technique for the Cuk converter (Figure 1) to limit the voltage deviation in response to a voltage step on the input.

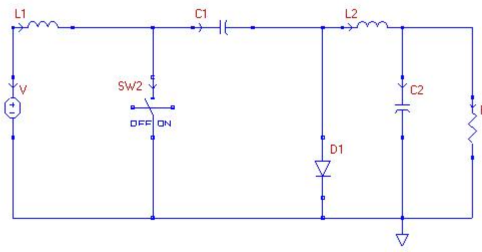

Figure 1: Cuk converter

In the following sections, the design requirements for this converter will be defined, followed by a derivation of a feedback model for the converter. The final section will describe two different modeling designs for the closed-loop Cuk converter, with the advantages and disadvantages discussed for each.

Cuk converter design requirements

For this particular design, the specifications are as follows: $V_g = 12V$, $V_o = -24V$, $f_s = 100kHz$, and $R_L = 12\Omega - 120\Omega$. The PWM conversion gain will be $\frac{1}{V_m} = \frac{1}{5}$, and the maximum acceptable output ripple for the system will be 5%. For this design it is also desired to keep the converter operating in Continuous Conduction Mode (CCM) and Continuous Voltage Mode (CVM) mode, as single quadrant switches are to be implemented. The input voltage step that will be applied at the load will stepped from 12 V to 20 V. Figure 2 shows the complete Cuk converter with the inclusion of the previously listed values.

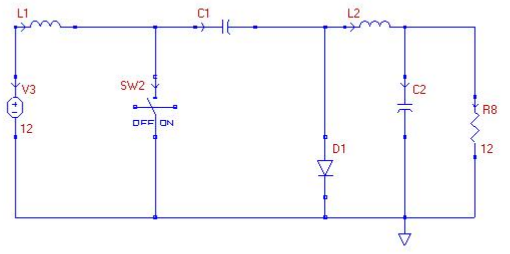

Figure 2: Cuk converter

Component selection based on design requirements

As described above, it is desired that this converter have an output ripple of less than 5% of the output voltage at steady state. Applying state space analysis for second order ripple, the output ripple for V_2 is found and is shown below. For reference, all DC and ripple qauntities are summarized below:

$$I_1 = \frac{V_g}{R}\frac{D^2}{D'^2}$$
$$I_2 = -\frac{V_g}{R}\frac{D}{D'}$$
$$V_1 = \frac{V_g}{D'}$$
$$V_2 = \frac{-V_g D}{D'}$$
$$\Delta i_1 = \frac{V_g}{L_1} D T_S$$
$$\Delta i_2 = \frac{V_g}{L_2} D T_S$$
$$\Delta v_1 = \frac{I_2}{C_1} D T_S$$
$$\Delta v_2 = \frac{T_S^2}{8}\frac{(V_1+V_2)D}{L_2 C_2}$$

Calculating $D = \frac{2}{3}$ for the desired output voltage, it is found that the first component limiting equation for the design is set by the voltage ripple equation, and is as shown below:

$$L_2 C_2 \geq 8\mu \tag{1}$$

The next limitation to the design is that the converter is required to operate in CCM. This requires that only positive currents flow through the diode.

$$\frac{\Delta i_{dpk-pk}}{2} < I_d \Rightarrow \frac{\Delta i_1 + \Delta i_2}{2} < I_1 + I_2$$

Rearranging the above equation and solving for L_1 yields the second component equation:

$$L_1 = \frac{L_2 R T_S {D'}^2}{2L_2 - R T_S D'^2} \tag{2}$$

The last component limiting requirement for the Cuk design is that the converter needs to operate in CVM. To achieve this, the steady state voltage ripple on C1 has to be less than the average voltage.

$$\frac{\Delta v_{1pk-pk}}{2} < V_1$$

This yields the final component constraint

$$C_1 > \frac{D^2 T_S}{2R} \tag{3}$$

All three of the above design constraints will be utilized later on when component values are chosen for the converter.

1) Modeling the Cuk converter

To design a compensation scheme to limit the change in output voltage in response to input disturbances, it is important to understand the structure of the converter control loop (shown in Figure 3). For this design, the load current is assumed constant and will have no effect on the output voltage. The main blocks to be designed are $G_{vd}(s)$ and $G_c(s)$, as they are the only sections of the control loop limited by the requirements outlined above. In the next section, the design of $G_{vd}(s)$ will be presented.

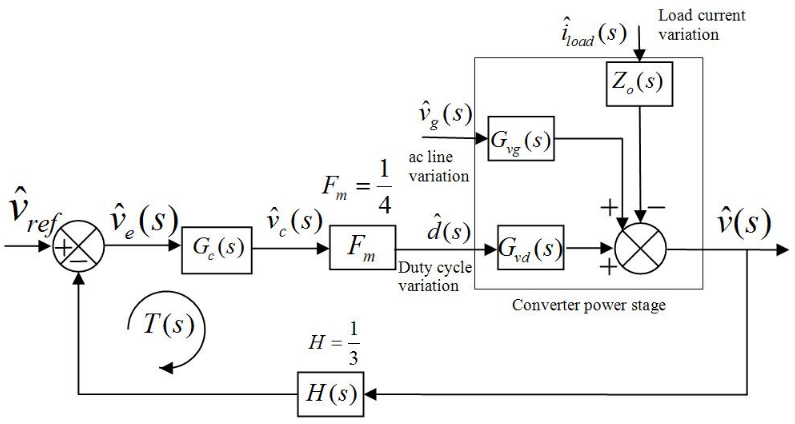

Figure 3: Cuk converter control loop

Derivation of the control to output transfer function

Listed below are the fundamental equations for state space analysis. x defines the state variables of the system and the u variables define the inputs. The number of states is defined by the number of storage elements in the system. For the Cuk converter, there are four states. The output voltage of the converter is the voltage across the capacitor C_2.

$$x = \begin{bmatrix} i_1 \\ i_2 \\ v_1 \\ v_2 \end{bmatrix}$$

$$u = v_g$$

$$y = v$$

$$\dot{x} = Ax + Bu$$
$$y = Cx + Eu$$

Applying circuit analysis techniques, the capacitor current and inductor voltage equations are found and summarized below for both switch positions.

During DTs,

$$\frac{di_1}{dt} = \frac{V_g}{L_1}$$
$$\frac{di_2}{dt} = \frac{v_1}{L_2} - \frac{v_2}{L_2}$$
$$\frac{dv_1}{dt} = -\frac{i_2}{C_1}$$
$$\frac{dv_2}{dt} = \frac{i_2}{C_2} - \frac{v_2}{RC_2}$$

During D'Ts,

$$\frac{di_1}{dt} = -\frac{v_1}{L_1} + \frac{V_g}{L_1}$$
$$\frac{di_2}{dt} = -\frac{v_2}{L_2}$$
$$\frac{dv_1}{dt} = \frac{i_1}{C_1}$$
$$\frac{dv_2}{dt} = \frac{i_2}{C_2} - \frac{v_2}{RC_2}$$

With the circuit defined over the two subintervals, the A, B, C, and E matrices can be defined as shown below:

$$A_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{L_2} & -\frac{1}{L_2} \\ 0 & -\frac{1}{C_1} & 0 & 0 \\ 0 & \frac{1}{C_2} & 0 & -\frac{1}{RC_2} \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 0 & -\frac{1}{L_1} & 0 \\ 0 & 0 & 0 & -\frac{1}{L_2} \\ \frac{1}{C_1} & 0 & 0 & 0 \\ 0 & \frac{1}{C_2} & 0 & -\frac{1}{RC_2} \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & -\frac{D'}{L_1} & 0 \\ 0 & 0 & \frac{D}{L_2} & -\frac{1}{L_2} \\ \frac{D'}{C_1} & -\frac{D'}{C_1} & 0 & 0 \\ 0 & \frac{1}{C_2} & 0 & -\frac{1}{RC_2} \end{bmatrix}$$

$$B_1 = B_2 = B = \begin{bmatrix} \frac{1}{L_1} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C_1 = C_2 = C = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

$$E_1 = E_2 = E = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

With the state space matrices defined, the control to output transfer function can be calculated as $G_{vd}(s) = C(sI - A)^{-1}Bd + Ed$, where $B_d = (A_1 - A_2)X + (B_1 - B_2)U$ and $E_d = (C_1 - C_2)X + (E_1 - E_2)U$. Applying basic matrix manipulation techniques, $G_{vd}(s)$ is calculated below. For a detailed calculation using MATLAB, please refer to Appendix 1.

$$X = -A^{-1}BU = \begin{bmatrix} \frac{V_g D^2}{RD'^2} \\ -\frac{V_g}{RDD'} \\ -\frac{V_g}{D'} \\ \frac{V_g}{DD'} \end{bmatrix}$$

$$B_d = (A_1 - A_2)X + (B_1 - B_2)U = \begin{bmatrix} -\frac{V_g}{L_1 D'} \\ -\frac{V_g}{L_2 D'} \\ -\frac{V_g}{C_1 R D'^2} \\ 0 \end{bmatrix}$$

$$E_d = (C_1 - C_2)X + (E_1 - E_2)U = 0$$

$$G_{vd}(s) = C(sI - A)^{-1}Bd + Ed =$$

$$\frac{V_g}{D'^2} \frac{s^2 \frac{L_1 C_1}{D'} - s\frac{D^2 L_1}{D'^2 R} + 1}{s^4 \frac{L_1 C_1 L_2 C_2}{D'^2} + s^3 \frac{L_1 C_1 L_2}{D'^2 R} + s^2 \left(\frac{L_1 C_1 L_2}{D'^2} + L_2 C_2 + \frac{D^2}{D'^2} L_1 C_2 \right) + s\left(\frac{L_2}{R} + \frac{D^2 L_1}{D'^2 R} \right) + 1}$$

Since $T(s) = G_{vd}(s) F_m G_c(s) H$, $G_{vd}(s)$ directly effects the loop gain of the system, and therefore its stability. Looking at the result above, it is not intuitive as to what the shape of the $G_{vd}(s)$ transfer function will look like, due to the fourth order polynomial in the denominator. In order to get a grasp on what the fourth order transfer function is doing, and to therefore be able to design with it, the transfer function needs to be factored.

Approximation 1: Cuks approximation

This section will describe the original factorization method developed for the Cuk converter, orignally presented by Slobodan Cuk. Essentially, it is an approximation that allows the fourth order denominator to by separated into two second order factors.

Shown below is Cuk's factorization of the denominator of the transfer function

$$\left(s^2\frac{L_1C_1}{D'^2}+s\frac{D^2L_1}{D'^2R}+1\right)\left(s^2L_2C_2+s\frac{L_2}{R}+1\right)$$

This is not an exact factorization of the fourth order polynomial. To determine the differences in the forms, the factorization has been expanded and is shown below:

$$s^4\frac{L_1C_1L_2C_2}{D'^2}+s^3\left(\frac{L_1C_1L_2}{D'^2R}+\frac{D^2L_1L_2C_2}{D'^2R}\right)+s^2\left(\frac{L_1C_1}{D'^2}+L_2C_2+\frac{D^2L_1L_2}{D'^2R^2}\right)+s\left(\frac{L_2}{R}+\frac{D^2L_1}{D'^2R}\right)+1$$

Notice the difference in the s^4 terms and the s^2 terms. The differences imply that for this to be a good approximation, the differences must be negligible. This happens when the following two conditions hold true. Condition one:

$$\frac{L_1C_1L_2}{D'^2R}\gg\frac{D^2L_1L_2C_2}{D'^2R}$$

Which reduces to

$$C_1\gg D^2C_2$$

Condition two: The actual s^2 term requires

$$\frac{L_1C_1}{D'^2}\gg\frac{D^2L_1C_2}{D'^2}$$

and the factored s^2 term requires

$$\frac{L_1C_1}{D'^2}\gg\frac{D^2L_1C_2}{D'^2R^2}$$

by multiplying each side by $\frac{D'^2}{D^2}\frac{1}{L}$, this reduces to

$$C_1\gg\frac{L_2D^2}{R^2}$$

Using Cuk's factorization shown at the beginning of the section, it can be seen that there is a pair of right half plane zeroes and also two pairs of poles as shown below.

$$\omega_z=\sqrt{\frac{D'}{L_1C_1}}$$

$$\omega_{p_1}=\frac{D'}{\sqrt{L_1C_1}}$$

$$\omega_{p_2}=\frac{1}{\sqrt{L_2C_2}}$$

Figures 4 - 6 show the three possible cases for the location of the poles and zeroes. As can be seen by examination, it is desirable to use the third case shown where $\omega_{p_2}\ll\omega_{p_1}\approx\omega_z$, because the phase is above -180 degrees for a larger range of frequencies.

Due to the factorization approximations listed above, the ideal pole placement is not possible because C_1 is required to be larger than L_2 multiplied by a constant and C_2 multiplied by a constant. If L_1 is a realistic value (which it has to be to meet the design requirements introduced previously) then ω_{p_1} and ω_z are smaller than ω_{p_2} due to this approximation. Designing with the third case would violate the approximation conditions and thereby make the factorization not hold true. To realize the desired graphical shape, a new approximation method is introduced in the following section.

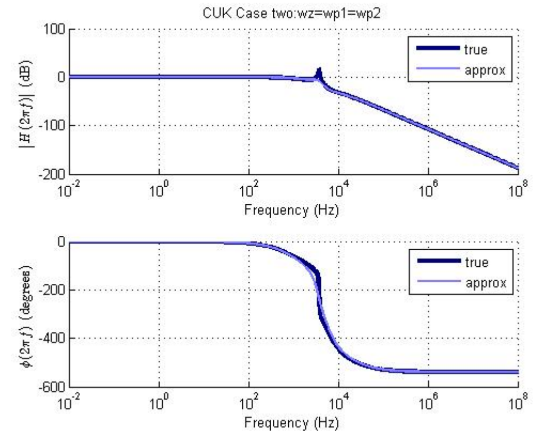

Figure 4: $\omega_{p_2} = \omega_{p_1} \approx \omega_z$

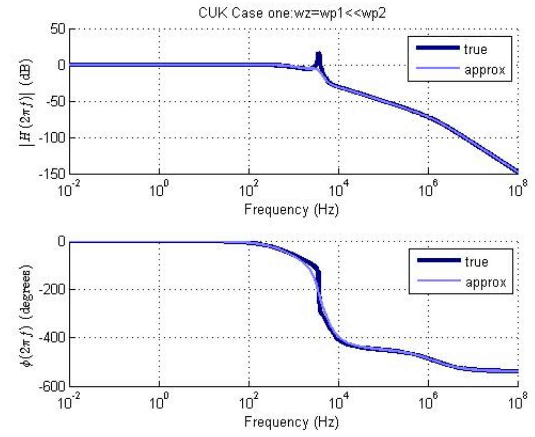

Figure 5: $\omega_{p_2} \gg \omega_{p_1} \approx \omega_z$

Approximation 2: Infinite Q

Another factorization is proposed below that is slightly different than Cuk's. As such, it will be examined in order to see if it can be utilized for a better design.

$$\left(s^2\frac{L_1C_2}{D'^2}+1\right)\left(s^2L_2C_2+s\frac{L_2}{R}+1\right)$$

To determine the necessary approximations for this factorization to be valid, it has been expanded below

$$= s^4\frac{L_1C_1L_2C_2}{D'^2}+s^3\frac{L_1C_1L_2}{D'^2R}+s^2\left(\frac{L_1C_1}{D'^2}+L_2C_2\right)+s\frac{L_2}{R}+1$$

Note that this is not an exact factorization of the actual transfer function denominator. The real function denominator is shown again as reference

$$s^4\frac{L_1C_1L_2C_2}{D'^2}+s^3\frac{L_1C_1L_2}{D'^2R}+s^2\left(\frac{L_1C_1L_2}{D'^2}+L_2C_2+\frac{D^2}{D'^2}L_1C_2\right)+s\left(\frac{L_2}{R}+\frac{D^2L_1}{D'^2R}\right)+1$$

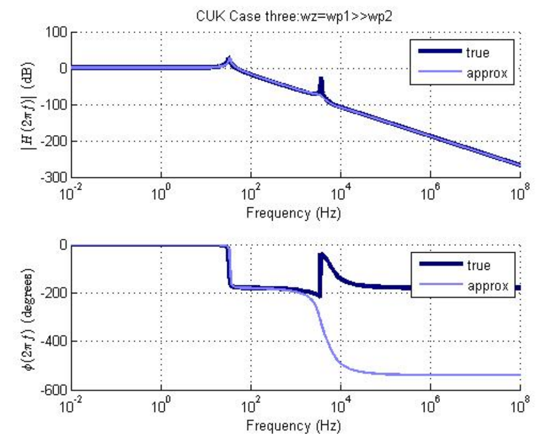

Figure 6: $\omega_{p_2} \ll \omega_{p_1} \approx \omega_z$

One of the requirements for this factorization to hold true to the original denominator is when the second s term in the real denominator is much less than the s term shown in the approximated denominator. This leads to the following condition:

$$\frac{D^2}{D'^2}\frac{L_1}{R} \ll \frac{L_2}{R}$$

which can be reduced to:

$$L_2 \gg \frac{D^2}{D'^2}L_1 \tag{4}$$

The other requirement is that the s^2 terms are approximately equal. This leads to the following equality:

$$\frac{D^2}{D'^2}L_1C_2 \ll \frac{L_1C_1}{D'^2} + L_2C_2$$

When looking at just L_2C_2 on the right side of the comparison, it is seen that this is the same requirement as before($L_2 \gg \frac{D^2}{D'^2}L_1$). Since the other term, $\frac{L_1C_1}{D'^2}$, is positive, it only eases the requirement and it is reasonable to drop it out to simplify the requirement. This causes only one requirement for this approximation to hold true. Seeing that there are no longer limitations on C_2 for this approximation, it is now possible to realize the desired response.

One problem with this design is that there is now an “infinite” Q in one of the denominator pole pairs. In order to limit peaking in the loop gain response and possibly a second crossover point, the following section will discuss methods to reduce this high Q.

2) G_{vd} damping methods

Lossless Damping

One approach to damping the Q is to add a capacitor in parallel to C_1 that is much larger. This large capacitor will have a resistance in series that will be modeled, but this will only affect the AC signals. At DC the large capacitor is seen as an open circuit and therefore there will be no loss through the series resistance with the large capacitor. This can be seen in Figure 7.

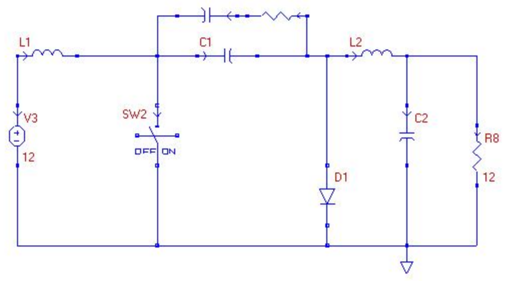

Figure 7: Lossless damped Cuk converter

To model this addition, the factored denominator of the control to output transfer function will be changed wherever there is a $\frac{1}{sC_1}$ term. The new large capacitor C and resistance r_c are in parallel with the C_1. The new control to output tranfer function approximation is found by expanding this relationship, and replacing the $\frac{1}{sC_1}$ term with the changes. This can be seen below.

$$\frac{1}{sC_1} \Rightarrow \frac{1}{sC_1} \left\| \left(\frac{1}{sC} + r_C \right) = \frac{s^2 C_1 C \left(\frac{1}{s^2 C_1 C} + \frac{r_c}{sC_1} \right)}{s^2 C_1 C \left(\frac{1}{sC_1} + \frac{1}{sC} + r_C \right)} \right.$$

And simplifying even further yields

$$\frac{1 + sCr_c}{s(C + C_1)1 + \left(s\frac{CC_1 r_c}{C + C_1} \right)}$$

Referring back to the factorization shown below, the sC_1 term can be replaced with the new addition.

$$\left(s^2 \frac{L_1 C_1}{D'^2} + 1 \right) \left(s^2 L_2 C_2 + s\frac{L_2}{R} + 1 \right)$$

$$\frac{s^2 L_1 C_1}{D'^2} + 1 \Rightarrow \frac{sL_1}{D'^2} \frac{s\left(C_1 + C\right)\left(1 + sr_c \frac{C_1 C}{C_1 + C}\right)}{1 + sr_c C} + 1$$

And expanding results with:

$$= \frac{1}{1 + sr_c C} \left[\frac{s^3 L_1 r_c C_1 C}{D'^2} + \frac{s^2 L_1 \left(C_1 + C\right)}{D'^2} + sr_c C + 1 \right]$$

A factorization of this leads to:

$$= \frac{1}{1 + sr_c C} \left[(sr_c C + 1) \left(\frac{s^2 L_1 C_1}{D'^2} + \frac{sL_1}{D'^2 r_c} + 1 \right) \right]$$

This factorization is not exact and is expanded below.

$$= \frac{1}{1 + sr_c C} \left[\frac{s^3 L_1 r_c C_1 C}{D'^2} + \frac{s^2 L_1 \left(C_1 + C\right)}{D'^2} + s\left(r_C C + \frac{L_1}{D'^2 r_C} \right) + 1 \right]$$

It can be seen that there is an extra term introduced in the factorization. This new term $\frac{sL_1}{D'^2 r_c}$ is the desired damping term. This factorization holds true when:

$$r_C C \gg \frac{L_1}{D'^2 r_C}$$

Which rearranges to:

$$L_1 \ll D'^2 r_c^2 C \tag{5}$$

The addition of the large capacitor and series resistance can now be used for the design using the damping term. However, since this damping term is being used in the factored control to output transfer function, a new state state space analysis will need to be performed to get the exact control to output transfer function.

Applying state space analysis to the circuit with added large capacitor and series resistance (shown in Figure 7) will give the new exact control to output transfer function to be used later in plotting the loop gain.

During DTs,

$$\frac{di_1}{dt} = \frac{V_g}{L_1}$$

$$\frac{di_2}{dt} = \frac{v_1}{L_2} - \frac{v_2}{L_2}$$

$$\frac{dv}{dt} = \frac{v_1}{rC} - \frac{v}{rC}$$

$$\frac{dv_1}{dt} = -\frac{i_2}{C_1} - \frac{v_1}{RC_1} + \frac{v}{rC1}$$

$$\frac{dv_2}{dt} = \frac{i_2}{C_2} - \frac{v_2}{RC_2}$$

During D'Ts,

$$\frac{di_1}{dt} = -\frac{v_1}{L_1} + \frac{V_g}{L_1}$$

$$\frac{di_2}{dt} = -\frac{v_2}{L_2}$$

$$\frac{dv}{dt} = \frac{v_1}{rC} - \frac{v}{rC}$$

$$\frac{dv_1}{dt} = \frac{i_1}{C_1} + \frac{v}{rC_1} - \frac{v_1}{rC_1}$$

$$\frac{dv_2}{dt} = \frac{v_2}{C_2} - \frac{v_2}{RC_2}$$

With the circuit defined over the two subintervals, the A, B, C, and E matrices can be defined as shown below:

$$A_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{L_2} & -\frac{1}{L_2} & 0 \\ 0 & -\frac{1}{C_1} & -\frac{1}{rC_1} & 0 & \frac{1}{rC_1} \\ 0 & \frac{1}{C_2} & 0 & -\frac{1}{RC_2} & 0 \\ 0 & 0 & \frac{1}{rC} & 0 & -\frac{1}{rC_1} \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 0 & 0 & -\frac{1}{L_1} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{L_2} & 0 \\ \frac{1}{C_1} & 0 & -\frac{1}{rC_1} & 0 & \frac{1}{rC_1} \\ 0 & \frac{1}{C_2} & 0 & -\frac{1}{RC_2} & 0 \\ 0 & 0 & \frac{1}{rC} & 0 & -\frac{1}{rC_1} \end{bmatrix}$$

$$A = A_1 D + A_2 D'$$

$$B_1 = B_2 = B = \begin{bmatrix} \frac{1}{L_1} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

With the state space matrices defined, the control to output transfer function can be calculated as $G_{vd}(s) = C(sI - A)^{-1}Bd + Ed$, where $B_d = (A_1 - A_2)X + (B_1 - B_2)U$ and $E_d = (C_1 - C_2)X + (E_1 - E_2)U$. Applying basic matrix manipulation techniques, $G_{vd}(s)$ is calculated below.

$$X = -A^{-1}BU = \begin{bmatrix} -I_o \\ DV_g + I_oR_l \end{bmatrix}$$

$$B_d = (A_1 - A_2)X + (B_1 - B_2)U = \begin{bmatrix} \frac{V_g}{L} \\ 0 \end{bmatrix}$$

$$E_d = (C_1 - C_2)X + (E_1 - E_2)U = 0$$

$$G_{vd}(s) = C(sI - A)^{-1}Bd + Ed$$

Refer to Appendix 2 for the MATLAB code on computing the exact control to output transfer function for the lossless damping case.

Parasitic Damping

Another approach to lowering the peaking with the factorization being used is to include the parasitic resistances associated with the componenets in the circuit. These real parasitics cause additional terms in the control to output transfer function that dampen the infinite Q. This approach has the advantage of not adding any additional components. It is only modeling the real parasitics of the components already in the circuit. This can be oberved in Figure 8.

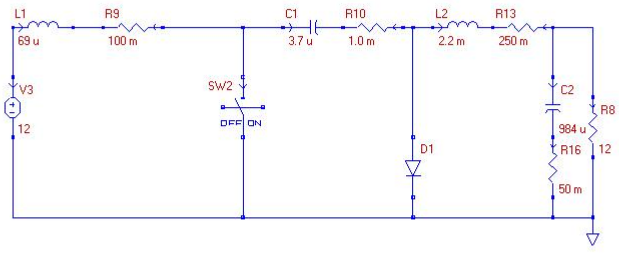

Figure 8: Parasitic damped Cuk converter

The exact control to output transfer function can be determined once again by applying state space analysis to the circuit with the parasitics added (Figure 8).

During DTs,

$$\frac{di_1}{dt} = -\frac{r_{L_1} i_1}{L_1} + \frac{V_g}{L_1}$$

$$\frac{di_2}{dt} = \frac{v_1}{L_2} - \frac{(r_{L_2} + r_{C_1} + r_{C_2} \| R)\, i_2}{L_2} - \left(\frac{r_{C_2}}{R + r_{C_2}} - 1 \right) \frac{v_2}{C_2}$$

$$\frac{dv_1}{dt} = -\frac{i_2}{C_1}$$

$$\frac{dv_2}{dt} = \frac{Ri_2}{(r_{C_2}+R)C_2} - \frac{v_2}{(R+r_{C_2})C_2}$$

During D'Ts,

$$\frac{di_1}{dt} = -\frac{(r_{L_1}+r_{C_1})\,i_1}{L_1} - \frac{v_1}{L_1} + \frac{V_g}{L_1}$$

$$\frac{di_2}{dt} = -\frac{(r_{L_2} + R\|\,r_{C_2})}{L_2} i_2 + \frac{\left(\frac{r_{C_2}}{R+r_{C_2}} - 1\right)}{L_2} v_2$$

$$\frac{dv_1}{dt} = \frac{i_1}{C_1}$$

$$\frac{dv_2}{dt} = \frac{Ri_2}{(r_{C_2}+R)C_2} - \frac{v_2}{(R+r_{C_2})C_2}$$

With the circuit defined over the two subintervals, the A, B, C, and E matrices can be defined as shown below:

$$A_1 = \begin{bmatrix} -\frac{r_{L_1}}{L_1} & 0 & 0 & 0 \\ 0 & -\frac{\left(r_{L_2}+r_{C_1}+r_{C_2}\|R\right)}{L_2} & \frac{1}{L_2} & -\left(\frac{r_{C_2}}{R+r_{C_2}} - 1\right)\frac{1}{C_2} \\ \frac{1}{C_1} & 0 & 0 & 0 \\ 0 & \frac{R}{(r_{C_2}+R)C_2} & 0 & -\frac{1}{(r_{C_2}+R)C_2} \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -\frac{r_{L_1}+r_{C_1}}{L_1} & 0 & -\frac{1}{L_1} & 0 \\ 0 & -\frac{\left(r_{L_2}+R\|r_{C_2}\right)}{L_2} & 0 & \frac{\left(\frac{r_{C_2}}{R+r_{C_2}}-1\right)}{L_2} \\ \frac{1}{C_1} & 0 & 0 & 0 \\ 0 & \frac{R}{(r_{C_2}+R)C_2} & 0 & -\frac{1}{(r_{C_2}+R)C_2} \end{bmatrix}$$

$$B_1 = B_2 = B = \begin{bmatrix} \frac{1}{L_1} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$A = A_1 D + A_2 D'$$

With the state space matrices defined, the control to output transfer function can be calculated as $G_{vd}(s) = C(sI-A)^{-1}Bd + Ed$, where $B_d = (A_1 - A_2)X + (B_1 - B_2)U$ and $E_d = (C_1 - C_2)X + (E_1 - E_2)U$. Applying basic matrix manipulation techniques, $G_{vd}(s)$ is calculated below.

$$X = -A^{-1}BU = \begin{bmatrix} -I_o \\ DV_g + I_o R_l \end{bmatrix}$$

$$B_d = (A_1 - A_2)X + (B_1 - B_2)U = \begin{bmatrix} \frac{V_g}{L} \\ 0 \end{bmatrix}$$

$$E_d = (C_1 - C_2)X + (E_1 - E_2)U = 0$$

$$G_{vd}(s) = C(sI - A)^{-1}Bd + Ed$$

Refer to Appendix 3 for the Matlab code on computing the exact control to output transfer function for the parasitic damping case.

Note that in this design there are too many component changes to accomplish a simple update of the denominator factorization, so it is a challenge to have a feel for what the response will look like. The next section will use the exact control to output transfer function to plot the loop gain.

3) Voltage Compensation Design

In the remainder of the report, a compensation scheme will be designed for the lossless case, and the results will be compared to that of the same compensator in the parasitic Cuk circuit. Finally, the two converter designs will be compared and it will be determined if the large capacitor of the lossless case is a necessary additional component for this design.

Lossless Damping Compensation

Now that a desirable form for $G_{vd}(s)$ has been derived for the lossless damping case, a compensation scheme can now be created to minimize the response of the system to disturbances in the input. Using the design limitations defined by equations 1-3 in MATLAB (see appendix 2), the component values are calculated to be $L1 = 68.7\mu H$, $L2 = 2.2mH$, $C1 = 3.7\mu F$, and $C2 = 984\mu F$. Using these values and making substitutions into equations 1-5, all the requirements and approximations have been met.

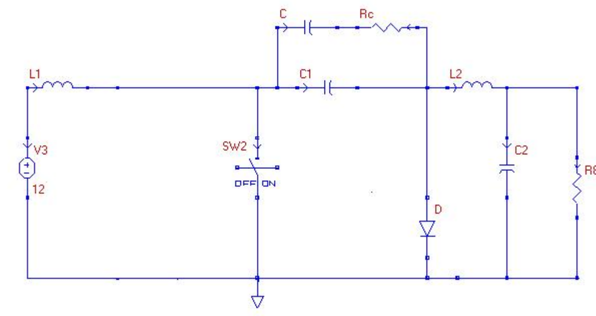

Figure 9: Lossless damping circuit

Figure 9 shows the circuit complete with the component values designed in MATLAB. Now that values have been chosen that meet the project requirements, a compensation scheme can be developed, using the fact that $T(s) = G_{vd}(s)F_mG_c(s)H$.

From Section one, it was found that it is desirable to have one of the pole pairs occur at a much lower frequency than the pole and zero pairs that are closely related to each other. In terms of compensation, it is desirable to make the loop gain such that the crossover frequency occurs after the first pole pair, but before the pole-zero pair combination. To accomplish this task, a PID compensator has been utilized(one pole at zero, two zeros). The integration term is desired for infinite DC gain, and the two zeros are necessary to increase the stability at the crossover frequency. Figure 9 shows the loop gain complete with the PID compensator. In Figure 11, please note the effectiveness of lossless damping to create a finite Q.

Now that a stable compensation loop has been designed for the Cuk converter with lossless damping, a simulation can be ran to determine the response to an input disturbance. Figure 12 shows the complete circuit designed using PECS, and the results for a step disturbance of 12V to 20V is shown in Figures 13

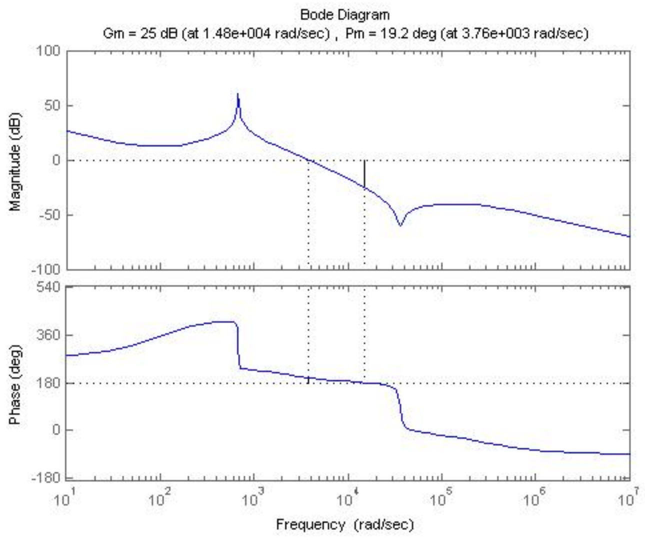

Figure 10: Lossless damping loop gain

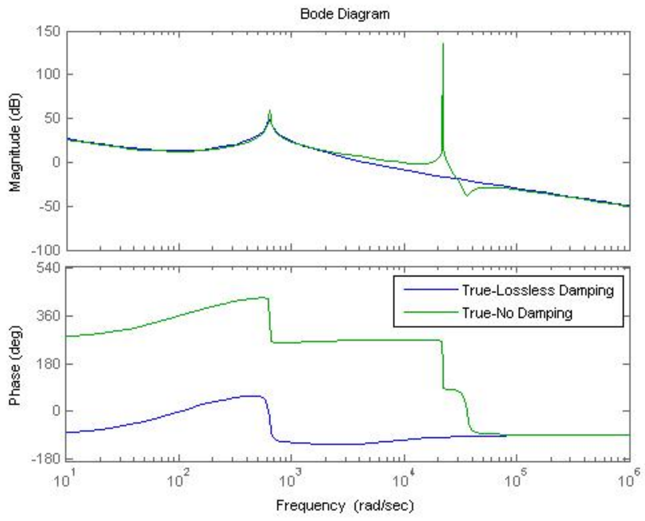

Figure 11: Comparison of lossless to basic Cuk

and 14 for each of the output load extremes. From the results of the input step response, it is easy to see that there is an undesirable characteristic in the response of the converter. The output responds correctly to a change in voltage from 12V to 20V correctly with approximately a 2V transient spike. However, when the input changes from 20V back to 12V, a very large spike appears on the output voltage. This spike occurs due to the system going into DCM mode for this large drop in input voltage. This does not mean that the circuit does not meet the initial specifications, as it was designed to operate in CCM for an input voltage of 12V. Rather, the large input disturbance from high to low causes the circuit to operate in DCM very briefly, before returning to CCM. If a two quadrant switch was implemented, this problem would not appear.

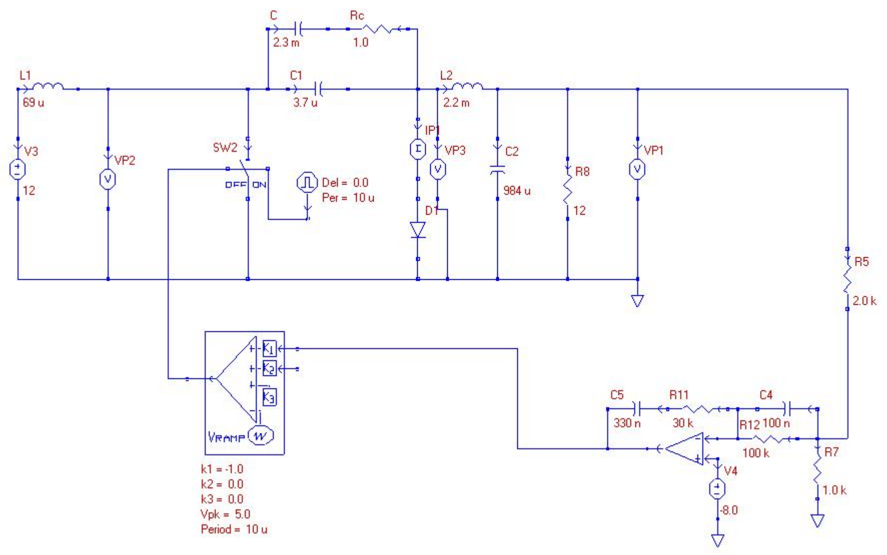

Figure 12: Lossless damping circuit

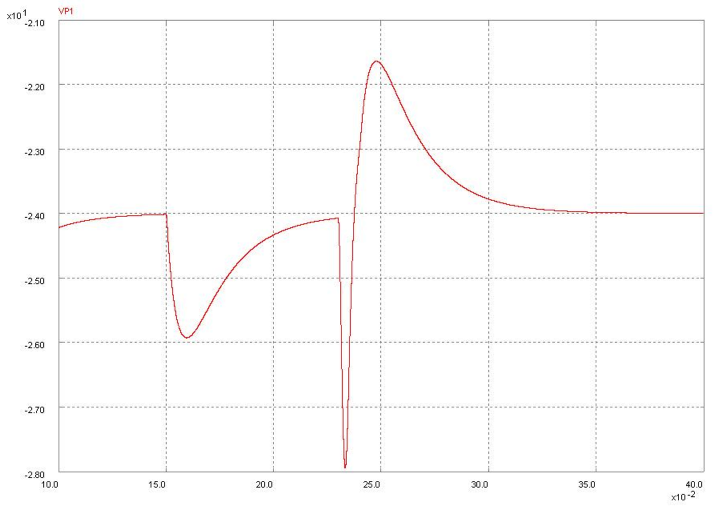

Figure 13: Output response to an input voltage step $R_L = 12\Omega$

In the following section, the same compensator that was designed for this circuit will be added to the Cuk converter with parasitics, and the response of the system will be compared to that of the lossless damping case.

Parasitic Damping Compensation

Figure 15 shows the Cuk converter with parasitics with realistic parasitic values for each of the components. As was explained in the previous section, the compensator will remain the same so the performance of the circuit can be compared directly to the previous case.

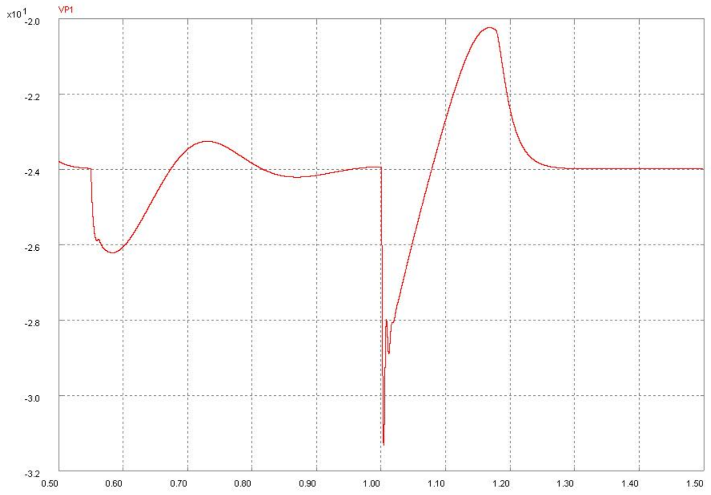

Figure 14: Output response to an input voltage step $R_L = 120\Omega$

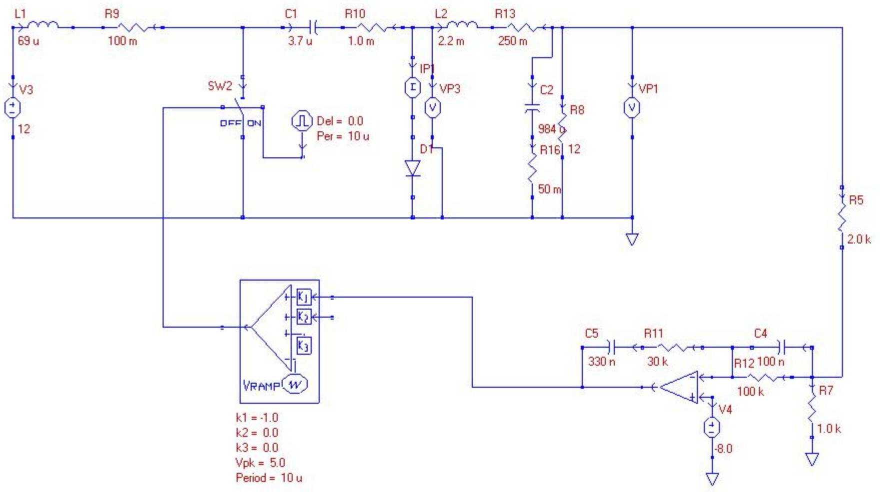

Figure 15: Cuk circuit including parasitics

In Figure 16, it is seen that the loop gain has an undesirable characteristic that the lossless damping case removed. At the location of pole with the high Q, the peaking causes the loop gain to cross over the 0dB point after the initial crossover. This is an undesirable trait because the phase at this point is beyond 180 degrees. However, it is a vast improvement compared to the Cuk converter without losses, as shown in Figure 17.

For comparison purposes, this circuit was simulated in PECS to compare the performance of the design to the lossless damping case. From Figures 18 and 19 it can be seen that the output voltage deviates slightly more than the lossless case, with a deviation of roughly 3.5V each direction. Also, looking at a close up view of the plot, a slight oscillation is visible, confirming that the peaking causes the circuit to be unstable.

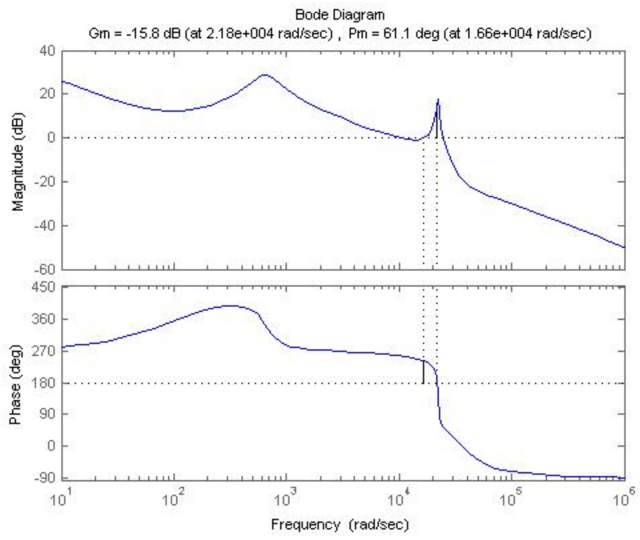

Figure 16: Cuk with parasitics loop gain

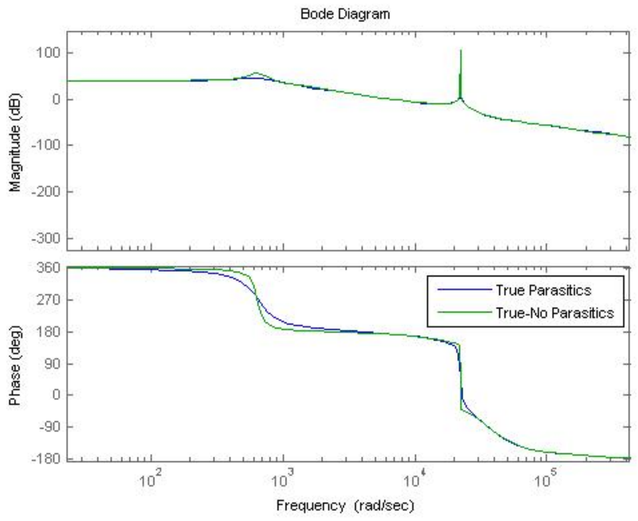

Figure 17: Comparison of Cuk with parasitics to the basic Cuk

Comparison of damping methods

From the examination shown above, it was found that adding a large capacitor to introduce finite peaking in the response of the circuit is a desirable trait. The Cuk with parasitics does reduce the Q, but not to a level low enough with realistic values of parasitic components. If the parasitic resistance on the L_1 inductor is increased to unrealistic levels, the Q can be reduced to a level below 0dB. However, a large increase in this resistance will also cause an increase in the output disturbance to a change in input voltage. By adding the additional capacitor to the circuit, lossless damping allows minimal output deviation to be achieved while still guaranteeing stability.

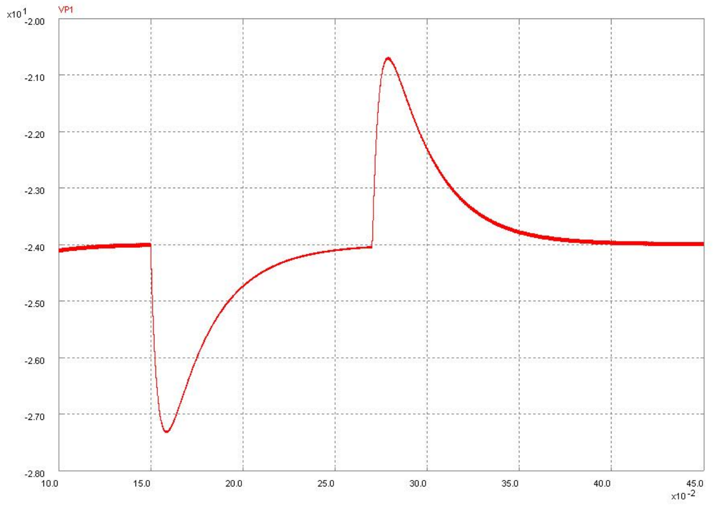

Figure 18: Output response to an input voltage step $R_L = 12\Omega$

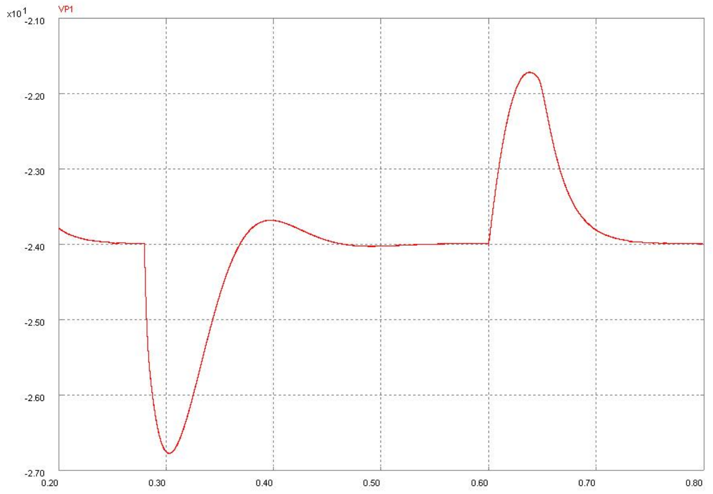

Figure 19: Output response to an input voltage step $R_L = 120\Omega$

Conclusion

In summary, through development of a small signal model for the control to output transfer function of the Cuk converter, a method for designing the converter components based on initial design constraints was formulated. By developing an approximation to the fourth order polynomial in the denominator of $G_{vd}(s)$, optimal placement of the poles and zeros was determined to optimize the stability of the loop gain. To reduce the peaking introduced by the design of the approximation, a lossless damping method was derived and showed effective damping of the undesired high Q shown in the plot of $G_{vd}(s)$.

By designing a PID compensator, output deviation in response to an 8V input step was reduced to approximately 2V for the lossless damping case. Compared to using only parasitic resistance to reduce the high peaking, lossless damping improves system stability and reduces the output response. Compared to the open loop, this method of compensation is an excellent choice for enviroments where input deviations are a significant issue.

Appendix 1: Matlab code for three implementations of the Cuk

```
close all;
clear all;
%==========================================================================
% Cuk Converter
% Andrew Chuinard, Tim Chairet
% 4/30/09
%==========================================================================
%--------------------------------------------------------------------------
% Parameters
%--------------------------------------------------------------------------
syms Vg L1 L2 C1 C2 D Dprime R s
syms i1 i2 v1 v2
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% Calculations
%--------------------------------------------------------------------------
xstate = [i1; i2; v1; v2];
% Matrices
A1 = [0 0 0 0; ...
      0 0 1/L2 1/L2; ...
      0 -1/C1 0 0; ...
      0 -1/C2 0 -1/(R*C2)];

A2 = [0 0 -1/L1 0; ...
      0 0 0 1/L2; ...
      1/C1 0 0 0; ...
      0 -1/C2 0 -1/(R*C2)];

B1 = [1/L1; 0; 0; 0];
B2 = [1/L1; 0; 0; 0];

C1 = [0 0 0 1];
C2 = [0 0 0 1];

E1 = 0;
E2 = 0;

A =  simplify(D*A1 + (1-D)*A2);
B =  simplify(D*B1 + (1-D)*B2);
C =  simplify(D*C1 + (1-D)*C2);
E =  simplify(D*E1 + (1-D)*E2);

X = simplify(-A^-1*B*Vg);

Bd = simplify((A1-A2)*X + (B1-B2)*Vg);

sI = [s 0 0 0; ...
```

```
        0 s 0 0; ...
        0 0 s 0; ...
        0 0 0 s];
Den = simplify(det(sI-A));
Den = collect(Den, s);

V_D = simplify(C*(sI-A)^(-1)*Bd);
simpleV_D = collect(V_D, s)
pretty(V_D)
pretty(simpleV_D)
%==================================================================
clc;
close all;
clear all;
%============================================================================
% Cuk Analysis
% Andrew Chuinard
% Tim Chairet
% xx/xx/09
%============================================================================

%----------------------------------------------------------------------------
% Initial Parameters
%----------------------------------------------------------------------------
Vg     = 12;
Vo      = -24;
Vo_ripple = .05;
fs      = 100e3;      % Switching Frequency
Rl1     = 12;       %Load Resistance minimum
Rl2     = 120;      %load resistance maximum
Vm     = 5;            % PWM Conversion Ratio
Fm     = 1/Vm;
Vg_step = 20;
D       = 2/3;
DeltaV = abs(.05*Vo);


%------------------------------------------------------------------------
% Initial component design meeting requirements
%------------------------------------------------------------------------
 L2 = 2.2e-3;  % Educated Guess
 C2 =2.599e4*(1/fs)^2*Vg*D/(8*L2*DeltaV);    % Ripple Requirement
 % DCM Mode  2(L1||L2)/(Rts)<D'^2  So design for biggest R
 %  L1>L2*Rl2*(1/fs)*(1-D)^2/2/(L2-Rl2*(1/fs)*(1-D)^2/2);
 L1=L2*Rl2*(1/fs)*(1-D)^2/2/(L2-Rl2*(1/fs)*(1-D)^2/2);
 %No DVM Requirement C1>D^2Ts/(2Rl1)  smallest R
 C1 =20*D^2*(1/fs)/(2*Rl1);

f       = logspace(-2,8, 10e3);
s       = j*2*pi*f;
%----------------------------------------------------------------------------
```

```
%===========================================================================
% Calculations
%===========================================================================


    %-----------------------------------------------------------------------
    % Case one:wz=wp1<<wp2
    %-----------------------------------------------------------------------
    % Factorization Requirements #1:  C1>> D^2C2  C1 >> (L2/R^2)D^2
    fprintf('Case one:wz=wp1<<wp2\n');
    % Determine wz, wp1, wp2
    wz                  = sqrt((1-D)/(L1*C1));
    wp1                 = (1-D)/sqrt(L1*C1);
    wp2                 = 1/sqrt(L2*C2);
    %Keep L2 the same
    C2 = 1/((wp1*10)^2*L2);    % To satisfy wp1, this fails the ripple requirement
    % Determine wz, wp1, wp2
    wz                  = sqrt((1-D)/(L1*C1))
    wp1                 = (1-D)/sqrt(L1*C1)
    wp2                 = 1/sqrt(L2*C2)
    if C1 >= 10*D^2*C2
        fprintf('1st factorization approx met\n');
    else
        fprintf('1st factorization approx not met\n');
    end

    if C1 >= 10*(L2/Rl1^2)*D^2
        fprintf('2nd factorization approx met\n\n');
    else
        fprintf('2nd factorization approx not met\n\n');
    end

    %True Control to Output
    numeratorReal1 = s.^2*L1*C1/(1-D)-s*(D^2*L1/((1-D)^2*Rl1))+1;
    denominatorReal1    = s.^4*L1*C1*L2*C2/(1-D)^2+ ...
                          s.^3*(L1*C1*L2/((1-D)^2*Rl1))+ ...
                          s.^2*(L1*C1/(1-D)^2+L2*C2+D^2/(1-D)^2*L1*C2)+ ...
                          s*(L2/Rl1+D^2/(1-D)^2*L1/Rl1)+1;
    Gvd_Real1           = numeratorReal1./denominatorReal1;

    Gvd_approx1         = numeratorReal1./((s.^2*L1*C1/(1-D)^2+s*D^2/(1-D)^2*L1/Rl1+1).* .
                          (s.^2*L2*C2+s*L2/Rl1+1));
    %-----------------------------------------------------------------------

    %-----------------------------------------------------------------------
    % Case two:wz=wp1=wp2
    %-----------------------------------------------------------------------
    % Determine L1,L2,C1,C2

    %Keep L2 the same
```

```
C2 = 1/(wp1^2*L2);   % To satisfy wp1, this fails the ripple requirement

fprintf('Case two:wz=wp1=wp2\n');
% Determine wz, wp1, wp2
wz                  = sqrt((1-D)/(L1*C1))
wp1                 = (1-D)/sqrt(L1*C1)
wp2                 = 1/sqrt(L2*C2)
% Factorization Requirements #1:  C1>> D^2C2  C1 >> (L2/R^2)D^2
if C1 >= 10*D^2*C2
    fprintf('1st factorization approx met\n');
else
    fprintf('1st factorization approx not met\n');
end

if C1 >= 10*(L2/Rl1^2)*D^2
    fprintf('2nd factorization approx met\n\n');
else
    fprintf('2nd factorization approx not met\n\n');
end

%True Control to Output
numeratorReal2 = s.^2*L1*C1/(1-D)-s*(D^2*L1/((1-D)^2*Rl1))+1;
denominatorReal2    = s.^4*L1*C1*L2*C2/(1-D)^2+ ...
                      s.^3*(L1*C1*L2/((1-D)^2*Rl1))+ ...
                      s.^2*(L1*C1/(1-D)^2+L2*C2+D^2/(1-D)^2*L1*C2)+ ...
                      s*(L2/Rl1+D^2/(1-D)^2*L1/Rl1)+1;
Gvd_Real2          = numeratorReal2./denominatorReal2;

Gvd_approx2         = numeratorReal2./((s.^2*L1*C1/(1-D)^2+s*D^2/(1-D)^2*L1/Rl1+1).* .
                      (s.^2*L2*C2+s*L2/Rl1+1));
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% Case three:wz=wp1>>wp2
%--------------------------------------------------------------------------
% Determine L1,L2,C1,C2

%Keep L2 the same
C2 = 1/((wp1/100)^2*L2);

fprintf('Case three:wz=wp1>>wp2\n');
% Determine wz, wp1, wp2
wz                  = sqrt((1-D)/(L1*C1))
wp1                 = (1-D)/sqrt(L1*C1)
wp2                 = 1/sqrt(L2*C2)
if (1-D)^2 >= 2*(L1*L2)/(L1+L2)/Rl2*fs
    fprintf('DCM\n');
else
    fprintf('CCM\n');
end
```

```
    % Factorization Requirements #1:  C1>> D^2C2  C1 >> (L2/R^2)D^2
    if C1 >= 10*D^2*C2
        fprintf('1st factorization approx met\n');
    else
        fprintf('1st factorization approx not met\n');
    end

    if C1 >= 10*(L2/Rl1^2)*D^2
        fprintf('2nd factorization approx met\n');
    else
        fprintf('2nd factorization approx not met\n');
    end

    %True Control to Output
    numeratorReal3 = s.^2*L1*C1/(1-D)-s*(D^2*L1/((1-D)^2*Rl1))+1;
    denominatorReal3    = s.^4*L1*C1*L2*C2/(1-D)^2+ ...
                          s.^3*(L1*C1*L2/((1-D)^2*Rl1))+ ...
                          s.^2*(L1*C1/(1-D)^2+L2*C2+D^2/(1-D)^2*L1*C2)+ ...
                          s*(L2/Rl1+D^2/(1-D)^2*L1/Rl1)+1;
    Gvd_Real3           = numeratorReal3./denominatorReal3;

    Gvd_approx3          = numeratorReal3./((s.^2*L1*C1/(1-D)^2+s*D^2/(1-D)^2*L1/Rl1+1).* .
                          (s.^2*L2*C2+s*L2/Rl1+1));
    %-------------------------------------------------------------------------


%--------------------------------------------------------------------------
% Plots
%--------------------------------------------------------------------------
magnitudeGvd_Real1          = 20*log10(abs(Gvd_Real1));
phaseGvd_Real1               = unwrap(angle(Gvd_Real1))*180/pi;
magnitudeGvd_approx1           = 20*log10(abs(Gvd_approx1));
phaseGvd_approx1              = unwrap(angle(Gvd_approx1))*180/pi;

magnitudeGvd_Real2          = 20*log10(abs(Gvd_Real2));
phaseGvd_Real2               = unwrap(angle(Gvd_Real2))*180/pi;
magnitudeGvd_approx2          = 20*log10(abs(Gvd_approx2));
phaseGvd_approx2             = unwrap(angle(Gvd_approx2))*180/pi;

magnitudeGvd_Real3          = 20*log10(abs(Gvd_Real3));
phaseGvd_Real3               = unwrap(angle(Gvd_Real3))*180/pi;
magnitudeGvd_approx3          = 20*log10(abs(Gvd_approx3));
phaseGvd_approx3             = unwrap(angle(Gvd_approx3))*180/pi;

figure;
subplot(2,1,1);
plotHandle = semilogx(f,magnitudeGvd_Real1,f,magnitudeGvd_approx1);
box off;
grid on;
set(plotHandle(1),'color',[0 0 0.5]);
set(plotHandle(2),'color',[0.5 0.5 1]);
set(plotHandle(1),'linewidth',3);
set(plotHandle(2),'linewidth',1.5);
xlabel('Frequency (Hz)');
ylabel('$|H(2 \pi f)|$ (dB)');
title('CUK Case one:wz=wp1<<wp2');
legend('true','approx');
set(get(gca,'ylabel'),'interpreter','LaTeX')

subplot(2,1,2);
    plotHandle = semilogx(f,phaseGvd_Real1,f,phaseGvd_approx1);
    box off;
    grid on;
    set(plotHandle(1),'color',[0 0 0.5]);
    set(plotHandle(2),'color',[0.5 0.5 1]);
    set(plotHandle(1),'linewidth',3);
    set(plotHandle(2),'linewidth',1.5);
    xlabel('Frequency (Hz)');
    ylabel('$\phi(2 \pi f)$ (degrees)');
    legend('true','approx');
    set(get(gca,'ylabel'),'interpreter','LaTeX');

figure;
subplot(2,1,1);
plotHandle = semilogx(f,magnitudeGvd_Real2,f,magnitudeGvd_approx2);
box off;
grid on;
set(plotHandle(1),'color',[0 0 0.5]);
set(plotHandle(2),'color',[0.5 0.5 1]);
set(plotHandle(1),'linewidth',3);
set(plotHandle(2),'linewidth',1.5);
xlabel('Frequency (Hz)');
ylabel('$|H(2 \pi f)|$ (dB)');
title('CUK Case two:wz=wp1=wp2');
legend('true','approx');
set(get(gca,'ylabel'),'interpreter','LaTeX')

subplot(2,1,2);
    plotHandle = semilogx(f,phaseGvd_Real2,f,phaseGvd_approx2);
    box off;
    grid on;
    set(plotHandle(1),'color',[0 0 0.5]);
    set(plotHandle(2),'color',[0.5 0.5 1]);
    set(plotHandle(1),'linewidth',3);
    set(plotHandle(2),'linewidth',1.5);
    xlabel('Frequency (Hz)');
    ylabel('$\phi(2 \pi f)$ (degrees)');
    legend('true','approx');
    set(get(gca,'ylabel'),'interpreter','LaTeX');

figure;
subplot(2,1,1);
```

```
plotHandle = semilogx(f,magnitudeGvd_Real3,f,magnitudeGvd_approx3);
box off;
grid on;
set(plotHandle(1),'color',[0 0 0.5]);
set(plotHandle(2),'color',[0.5 0.5 1]);
set(plotHandle(1),'linewidth',3);
set(plotHandle(2),'linewidth',1.5);
xlabel('Frequency (Hz)');
ylabel('$|H(2 \pi f)|$ (dB)');
title('CUK Case three:wz=wp1>>wp2');
legend('true','approx');
set(get(gca,'ylabel'),'interpreter','LaTeX')

subplot(2,1,2);
    plotHandle = semilogx(f,phaseGvd_Real3,f,phaseGvd_approx3);
    box off;
    grid on;
    set(plotHandle(1),'color',[0 0 0.5]);
    set(plotHandle(2),'color',[0.5 0.5 1]);
    set(plotHandle(1),'linewidth',3);
    set(plotHandle(2),'linewidth',1.5);
    xlabel('Frequency (Hz)');
    ylabel('$\phi(2 \pi f)$ (degrees)');
    legend('true','approx');
    set(get(gca,'ylabel'),'interpreter','LaTeX');
```

Appendix 2: Matlab code for lossless damping

```
close all;
clear all;
%==========================================================================
% 2nd Cuk Factorization Lossless Damping Case
% Andrew Chuinard
% Tim Chairet
% 05/24/09
%==========================================================================
%--------------------------------------------------------------------------
% Initial Parameters
%--------------------------------------------------------------------------
Vg      = 12;
Vo      = -24;
Vo_ripple = .05;
fs      = 100e3;      % Switching Frequency
Rl1     = 12;      %Load Resistance minimum
Rl2     = 120;     %load resistance maximum
Vm      = 5;           % PWM Conversion Ratio
Fm      = 1/Vm;
Vg_step = 20;
DeltaV = abs(.05*Vo);
%Damping
C = 2.3e-3;
```

```
r = 1;
D       = 2/3;
s = tf('s');
%-----------------------------------------------------------------------------
%=============================================================================
% Calculations
%=============================================================================
    %-------------------------------------------------------------------------
    % wz=wp1>wp2
    %-------------------------------------------------------------------------
 L2 = 2.2e-3;  % Educated Guess
 C2 =2.599e4*(1/fs)^2*Vg*D/(8*L2*DeltaV);    % Ripple Requirement
 % DCM Mode  2(L1||L2)/(Rts)<D'^2  So design for biggest R
 %  L1>L2*Rl2*(1/fs)*(1-D)^2/2/(L2-Rl2*(1/fs)*(1-D)^2/2);
 L1=L2*Rl2*(1/fs)*(1-D)^2/2/(L2-Rl2*(1/fs)*(1-D)^2/2);
 %No DVM Requirement  C1>D^2Ts/(2Rl1)   smallest R
 C1 =20*D^2*(1/fs)/(2*Rl1);
 % Determine wz, wp1, wp2
 wz                   = sqrt((1-D)/(L1*C1));
 wp1                  = (1-D)/sqrt(L1*C1);
 wp2                  = 1/sqrt(L2*C2);
    %Ripple Criteria
    if DeltaV >= (1/fs)^2*Vg*D/(8*L2*C2)
        fprintf('Ripple Criteria MET\n');
    else
        fprintf('Ripple Criteria NOT MET\n');
    end

    %CCM Criteria
    if (1-D)^2 >= 2*(L1*L2)/(L1+L2)/Rl2*fs
        fprintf('DCM \n');
    else
        fprintf('CCM \n');
    end

        %DVM Criteria
    if C1 >= D^2*(1/fs)/(2*Rl1);
        fprintf('CVM\n');
    else
        fprintf('DVM\n');
    end

    % Factorization Requirements
    if L2 >= 7.5*D^2/(1-D)^2*L1
        fprintf('Factorization approx met \n');
    else
        fprintf('Factorization approx not met\n');
    end

    % Lossless Factorization Requirements
    if L1 <= 10*(1-D)^2*r^2*C
        fprintf('Lossless Factorization approx met \n');
    else
        fprintf('Lossless Factorization approx not met\n');
    end

    % Print output ripple values
    i1Ripple = Vg/L1*D*(1/fs)
    i2Ripple = Vg/L2*D*(1/fs)
    V1Ripple = Vg*(1/fs)/(Rl1*C1)*(D/(1-D))^2
    V2Ripple = (1/fs)^2*Vg*D/(8*L2*C2)
    I1 = Vg/Rl2*(D/(1-D))^2
    I2 = Vg/Rl2*(D/(1-D))
    V1 = Vg/(1-D)
    V2 = Vg*(D/(1-D))

    % Control to Output
    numeratorReal = s^2*L1*C1/(1-D)-s*(D^2*L1/((1-D)^2*Rl2))+1;

    denominatorReal     = s^4*L1*C1*L2*C2/(1-D)^2+ ...
                            s^3*(L1*C1*L2/((1-D)^2*Rl2))+ ...
                            s^2*(L1*C1/(1-D)^2+L2*C2+D^2/(1-D)^2*L1*C2)+ ...
                            s*(L2/Rl2+D^2/(1-D)^2*L1/Rl2)+1;
    Gvd_Real            = numeratorReal/denominatorReal;


    Gvd_approx          = numeratorReal/((s^2*L1*C1/(1-D)^2+s*L1/((1-D)^2*r)+1)* ...
                            (s^2*L2*C2+s*L2/Rl2+1));

    %-----------------------------------------------------------------------------------
    % lossless control to output
    %-----------------------------------------------------------------------------------

    A1 = [0 0 0 0 0; ...
       0 0 (1/L2) (-1/L2) 0; ...
       0 (-1/C1) (-1/(r*C1)) 0 (1/(r*C1)); ...
       0 (1/C2)  0 (-1/(Rl2*C2)) 0; ...
       0 0 (1/(r*C)) 0 (-1/(r*C))];
    A2 = [0     0     -1/L1     0              0; ...
          0     0     0         -1/L2          0; ...
          1/C1  0     -1/(r*C1) 0              1/(r*C1); ...
          0     1/C2  0         -1/(Rl2*C2)    0; ...
          0     0     1/(r*C)   0              -1/(r*C);];
    B1 = [1/L1 ; 0 ; 0 ; 0 ; 0];
    B2 = [1/L1 ; 0 ; 0 ; 0 ; 0];
    Cm1 = [0 0 0 1 0];
    Cm2 = Cm1;
    E1 = [0];
    E2 = E1;
    A =  simplify(D*A1 + (1-D)*A2);
    B =  simplify(D*B1 + (1-D)*B2);
    Cm =  simplify(D*Cm1 + (1-D)*Cm2);
```

```
    E =  simplify(D*E1 + (1-D)*E2);

    X = -A^(-1)*B*Vg;

    Bd = (A1-A2)*X + (B1-B2)*Vg;
    Ed = (Cm1-Cm2)*X + (E1-E2)*Vg;
    sI = [s 0 0 0 0; ...
          0 s 0 0 0; ...
          0 0 s 0 0; ...
          0 0 0 s 0;
          0 0 0 0 s];

  sys = ss(A,B,Cm,E);
  V_G = tf(sys);
  sys2 = ss(A,Bd,Cm,Ed);
  V_D1 = tf(sys2);
  V_D = Cm*(sI-A)^(-1)*Bd + Ed;

% Compensator
Gcon = 10/s*(s/100+1)*(s/100+1);
%-------------------------------------------------------------------------
% Plots
%-------------------------------------------------------------------------
figure;
bode(Fm*V_D*Gcon,Fm*Gvd_Real*Gcon);
legend('True-Lossless Damping','True-No Damping');

figure;
 bode(V_G/(1+Gcon*Fm*V_D));
  title('Closed Loop Audiosusceptability')
figure;
margin(Fm*V_D*Gcon);
```

Appendix 3: Matlab code for parasitic damping

```
close all;
clear all;
%=========================================================================
% 2nd Cuk Factorization
% Andrew Chuinard
% Tim Chairet
% xx/xx/09
%=========================================================================

%-------------------------------------------------------------------------
% Initial Parameters
%-------------------------------------------------------------------------
Vg     = 12;
Vo      = -24;
Vo_ripple = .05;
fs     = 100e3;      % Switching Frequency
```

```
 Rl1     = 12;      %Load Resistance minimum
 Rl2     = 120;     %load resistance maximum
Vm     = 5;           % PWM Conversion Ratio
Fm     = 1/Vm;
 Vg_step = 20;
 DeltaV = abs(.05*Vo);

%Parasitics
 rl1 = 100e-3;
 rl2 = 250e-3;
 rc1 = 1e-3;
 rc2 = 50e-3;

D        = 2/3;
 DeltaV = abs(.05*Vo);

 s = tf('s');


%===============================================================================
% Calculations
%===============================================================================



  %----------------------------------------------------------------------------
  % wz=wp1>wp2
  %----------------------------------------------------------------------------


  L2 = 2.2e-3;  % Educated Guess
  C2 =2.599e4*(1/fs)^2*Vg*D/(8*L2*DeltaV);    % Ripple Requirement
  % DCM Mode  2(L1||L2)/(Rts)<D'^2  So design for biggest R
  %  L1>L2*Rl2*(1/fs)*(1-D)^2/2/(L2-Rl2*(1/fs)*(1-D)^2/2);
  L1=L2*Rl2*(1/fs)*(1-D)^2/2/(L2-Rl2*(1/fs)*(1-D)^2/2);
  %No DVM Requirement  C1>D^2Ts/(2Rl1)   smallest R
  C1 =20*D^2*(1/fs)/(2*Rl1);

     % Determine wz, wp1, wp2
wz                    = sqrt((1-D)/(L1*C1));
wp1                   = (1-D)/sqrt(L1*C1);
wp2                   = 1/sqrt(L2*C2);



    %Ripple Criteria
      if DeltaV >= (1/fs)^2*Vg*D/(8*L2*C2)
          fprintf('Ripple Criteria MET\n');
      else
          fprintf('Ripple Criteria NOT MET\n');
      end
```

```
%CCM Criteria
if (1-D)^2 >= 2*(L1*L2)/(L1+L2)/Rl2*fs
    fprintf('DCM \n');
else
    fprintf('CCM \n');
end

    %DVM Criteria
if C1 >= D^2*(1/fs)/(2*Rl1);
    fprintf('CVM\n');
else
    fprintf('DVM\n');
end

% Factorization Requirements
if L2 >= 7.5*D^2/(1-D)^2*L1
    fprintf('Factorization approx met \n');
else
    fprintf('Factorization approx not met\n');
end


% Print output ripple values
i1Ripple = Vg/L1*D*(1/fs)
i2Ripple = Vg/L2*D*(1/fs)
V1Ripple = Vg*(1/fs)/(Rl1*C1)*(D/(1-D))^2
V2Ripple = (1/fs)^2*Vg*D/(8*L2*C2)
I1 = Vg/Rl2*(D/(1-D))^2
I2 = Vg/Rl2*(D/(1-D))
V1 = Vg/(1-D)
V2 = Vg*(D/(1-D))





%True Control to Output
numeratorReal = s^2*L1*C1/(1-D)-s*(D^2*L1/((1-D)^2*Rl1))+1;

denominatorReal     = s^4*L1*C1*L2*C2/(1-D)^2+ ...
                        s^3*(L1*C1*L2/((1-D)^2*Rl1))+ ...
                        s^2*(L1*C1/(1-D)^2+L2*C2+D^2/(1-D)^2*L1*C2)+ ...
                        s*(L2/Rl1+D^2/(1-D)^2*L1/Rl1)+1;
Gvd_Real            = numeratorReal/denominatorReal;


Qp2 =1/(L2/Rl1*wp2);
%-----------------------------------------------------------------------------
A1 = [-(rl1/L1) 0 0 0 ; ...
    0 -(rc1+rl2+rc2*Rl1/(rc2+Rl1))/L2 (1/L2) (rc2/(Rl1+rc2)-1)/L2; ...
```

```
        0 (-1/C1) 0 0 ; ...
        0 (Rl1/(rc2+Rl1)/C2) 0 -(1/(rc2+Rl1)/C2)];

A2 = [-(rl1+rc1/L1) 0 -(1/L1) 0 ; ...
        0 -(rl2+rc2*Rl1/(rc2+Rl1))/L2 0 (rc2/(Rl1+rc2)-1)/L2; ...
        (1/C1) 0 0 0 ; ...
        0 (Rl1/(rc2+Rl1)/C2) 0 -(1/(rc2+Rl1)/C2)];


B1 = [1/L1 ; 0 ; 0 ; 0];
B2 = [1/L1 ; 0 ; 0 ; 0];

Cm1 = [0 0 0 1];
Cm2 = Cm1;

E1 = [0];
E2 = E1;


A =  simplify(D*A1 + (1-D)*A2);
B =  simplify(D*B1 + (1-D)*B2);
Cm =  simplify(D*Cm1 + (1-D)*Cm2);
E =  simplify(D*E1 + (1-D)*E2);


X = -A^(-1)*B*Vg;

Bd = (A1-A2)*X + (B1-B2)*Vg;
Ed = (Cm1-Cm2)*X + (E1-E2)*Vg;
sI = [s 0 0 0; ...
        0 s 0 0; ...
        0 0 s 0; ...
        0 0 0 s];
V_D = Cm*(sI-A)^(-1)*Bd + Ed;

sys = ss(A,B,Cm,E);
  V_G = tf(sys);
  sys2 = ss(A,Bd,Cm,Ed);
  V_D1 = tf(sys2);
% Compensator
Gcon = 10/s*(s/100+1)*(s/100+1);




%---------------------------------------------------------------------------
% Plots
%---------------------------------------------------------------------------
figure;
bode(V_D, Gvd_Real*100);
legend('True Parasitics','True-No Parasitics','k');
```

```
figure;
margin(Gcon*Fm*V_D);

figure;
margin(V_G/(1+Gcon*Fm*V_D));
title('Closed Loop Audiosusceptability')
```