EEEE2046 Electrical Project

Control System Design and Control System Modelling

1 Introduction

In the EEEE2046 “energy” project, you should design and implement closed loop control for your power supply. The control is needed to achieve two main aims:

- Steady state accuracy of the output voltage (for example when the load resistance is changed).
- Stopping (most of) the 100Hz ripple created by the rectifier from appearing at the power supply output.

In achieving these aims it is also a requirement that the control loop has good stability and transient response. Some of these ideas were discussed in the “Introduction to control lecture” – slides for that are on Moodle.

To do a control design (of anything) there are basically two tasks:

a) System modelling – to get a mathematical model of the system than can be used to do the control design.
b) Controller design - determining what type of controller to use and designing its parameters based on the mathematical model of the system and the control requirements.

For power electronic systems, (a) can be quite challenging because of the nature of the system (changing circuit topology when the switches operate, non-linearities etc) whilst there are often standard approaches that can guide us in (b). Fortunately, the Forward Converter is one of the easiest converters to model and control – one of the main reasons why we chose it for this project instead of something else (the Flyback converter for example).

This handout guides you through the process of designing and implementing a closed loop controller for your converter.

There are a number of appendices. Appendix A deals with the system modelling – its purpose is to provide the models needed to do the controller design. There will be some ideas here that are new to you (and are studied in more depth in years 3/4) – you should be able to understand most if not all of it – but if you find it difficult it is not essential to follow absolutely everything there – you really just need the models to do the design (which you do need to understand fully).

Appendix B deals with transfer function calculation – firstly for the output filter and secondly for calculating the parameters of the amplifier circuit to implement a particular controller design transfer function. Again – the results here are what you need to do the design – but you should be able to follow where they come from.

Appendices A and B are quite detailed – the key concepts and results that you will need to do your control design are highlighted in boxes

The main part of the handout deals with the controller design. This builds directly on ideas that you will have met in the control part of EEEE2045 and you will need to study it and understand it well to design a satisfactory controller.

2 Control Design

2.1 Initial Considerations

A reminder of the overall system from the main project handout

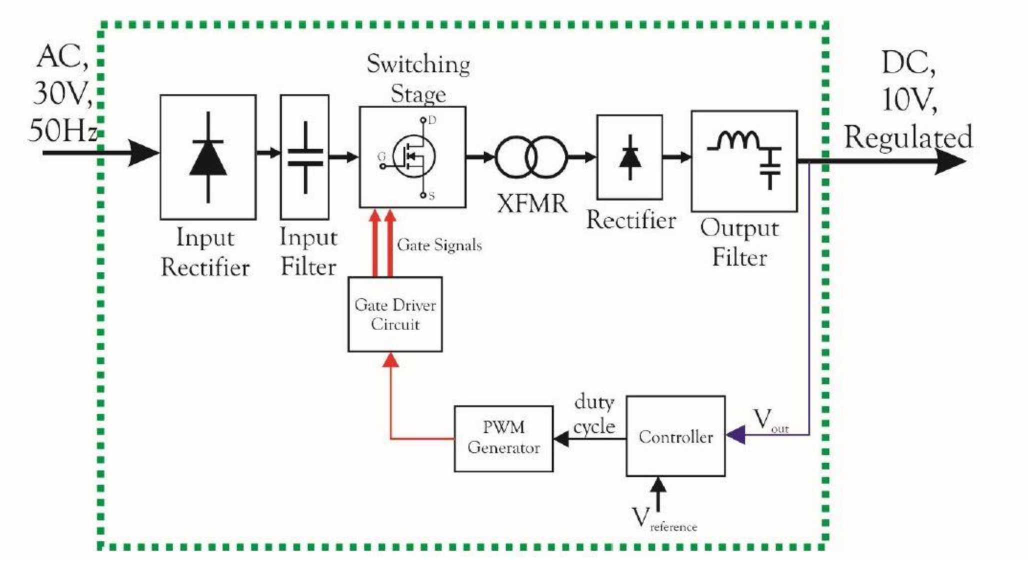


Fig 1 Overall SMPS system

To do the control design – we need to get a mathematical model of this system in a block diagram form that we can use with standard control design methods. That is not particularly easy – all the detail is given in Appendix A – but the resulting model, shown in Figure 2 is fortunately quite simple (it is a unity gain feedback system of the type you will have looked at in EEEE2045).

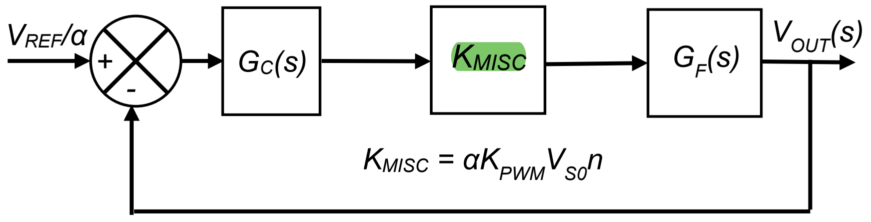


Figure 2 Block Diagram Model for Controller Design

In this model, $G_F(s)$ is the transfer function of the output filter, K_{MISC} is just a gain that includes a number of things (transformer turns ratio, PWM chip etc – see Appendix A) and $G_C(s)$ is the transfer function of the controller.

To do the control design we are going to use the Bode approach (because you have looked at it in EEEE2045 and it is a method that is frequently used in industry for this sort of design). Firstly, let's have a reminder of some things you have seen before – which we are going to use and which you will need to be very familiar with.

2.2 A Reminder about Bode, Transfer Functions and Block Diagrams

Here are a few definitions and mathematical operations we are going to need to use.

"Loop Gain" – the gain of all the blocks in the loop multiplied together. In the system of Fig 2 that we will consider, this is given by: loop gain = $K_{MISC}G_C(s)G_F(s)$.

"Crossover Frequency (ω_C) – this is the frequency at which the **magnitude** of the loop gain is unity (i.e = 1).

"Phase Margin" – this is the difference between -180^{O} and the phase shift of the loop gain at the crossover frequency.

"Gain Margin" – this is (1 – magnitude of loop gain) at the frequency where the loop phase shift is 180^{O} (it is the difference between the gain we have and the gain that would make the system unstable).

In order to use these definitions, we will need to calculate the gain magnitude and phase shift of blocks, defined as a Laplace transfer functions, at particular frequencies. Remember that to do this we simply replace s by $j\omega$ and then calculate the modulus (= gain magnitude) and angle (= phase shift) of the resulting complex number.

MATLAB provides a number of tools for manipulating transfer functions and for evaluating gain, phase and drawing Bode plots etc which will be useful – see Appendix C.

2.3 Controller Design – Venable Approach

The two main tasks in controller design are:

a) Deciding what the controller transfer function $G_C(s)$ should look like (how many poles and zeros etc).
b) Working out the parameters of the chosen transfer function to get the desired control performance.

For complicated control problems, (a) can be quite difficult and often a lot of experience is needed – we are lucky however as plenty of people have looked at the same problem before and there are well documented approaches we can use. To do (b), there are standard methods – the Bode approach is one of them.

For this project, we are going to use an approach for (a) based on a classic paper by "H Dean Venable":

Venable, H. Dean. "The K Factor: A New Mathematical Tool for Stability Analysis and Synthesis." Proc. Powercon 10. 1983. San Diego, CA. pp. H1−1 to H1−12

Venable's original paper is provided on Moodle for interest and reference – read it if you like but beware that some of his terminology and definitions are a little unusual and are potentially confusing. Everything you need to know about the method to do the design is in this handout.

Venable defines 3 alternative transfer functions for $G_C(s)$ (Type 1, Type 2, Type 3) of increasing complexity and provides the method for designing them to get adequate stability (phase margin). The 3 alternatives are discussed below – for your design it is probably a good idea to do a Type 1 design first to get something working – then, if needed and you have time, you can try and improve it using one of the other types. The method is explained below, and then a step by step procedure is given.

2.3.1 *A Simple Controller (Type 1)*

The transfer function for the Type 1 controller is a simple integrator. Remember that we want a very small steady state error – so we need an integrator in the controller (see EEEE2045 notes).

$$G_C(s) = \frac{A}{s}$$

There is only one quantity to choose here (*A*) so the design is quite simple. It is useful to define the gain of $G_C(s)$ at the crossover frequency:

$$G = |G_C(s)|_{\omega_c} = |G_C(j\omega_C)| = \left|\frac{A}{j\omega_C}\right| = \frac{A}{\omega_C} \quad (1)$$

Note that the phase shift of G_C is always 90^O (lagging) at all frequencies.

Following the Bode method we need to decide where to put the crossover frequency (ω_C) and then we need to work out *A* and check the phase and gain margins (to see how stable the system is).

Firstly, we need to consider the Bode plot of the filter transfer function $G_F(s)$ (see Appendix B.1 for the transfer function derivation). This will look something like below (**note – this is just an example for some specific parameters I made up – yours will be different – you will need to calculate it – you can't use this one!**):

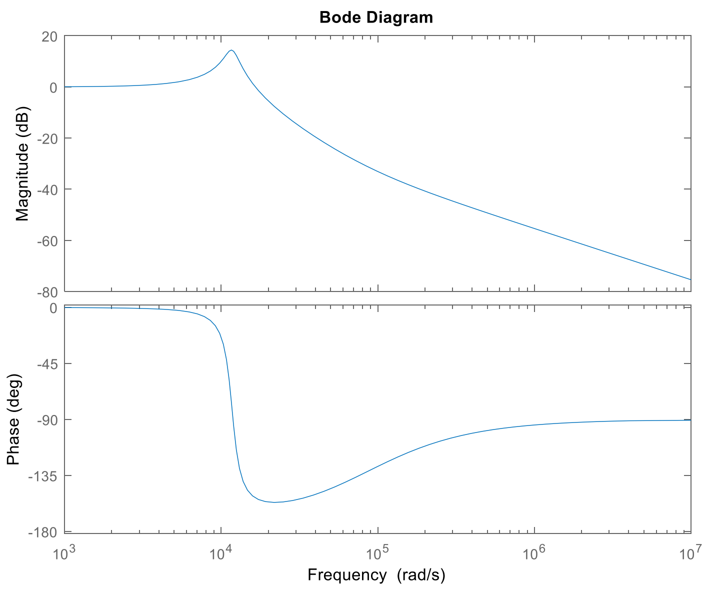


Figure 3 – Typical Bode plot for filter transfer function (G_F)

Some things to note about the filter characteristic:

- The magnitude response at low frequencies is 0dB (i.e unity) – low frequencies pass straight through the filter
- It has a peak in the magnitude response at the resonant frequency (about 1.9kHz, 12krad/s in this case – yours will be different!)
- After the peak, the magnitude reduces with frequency – this is why it filters the switching frequency components
- The phase shift at the resonant frequency is 90^O (lagging)
- The phase shift approaches 180^O (lagging) – as we expect from an LC filter – but then reduces back to 90^O at high frequencies. This is due to the zero created by the capacitor ESR.

We can immediately see that we have a problem here for this simple controller. There is a peak in the magnitude response at the filter resonant frequency and also, at this frequency, the loop phase shift will be 180^O (90^O from the integrator and 90^O from the filter) – the loop gain must be significantly less than unity at this frequency, or we will have no gain margin.

From above, it should be clear that the crossover frequency must be significantly less than the resonant frequency of the filter to get a system with reasonable stability (with this type of controller). There is no definitive approach to selecting ω_C however, and the "best" choice of ω_C (and hence A) will depend on your filter characteristic and what gain/phase margin you need.

I suggest that you start by setting the crossover frequency at 1/10 of the filter resonant frequency – this should give a "safe" design and one that you can get working in the lab.

There is a little extra complication because the gain K_{MISC} (see Fig 2) depends on the input voltage and the transfer function of the filter $G_F(s)$ depends on the load resistance – both of which are variable. To get around this, we have to decide what the "worst case" conditions are (the ones that will give minimum phase margin) and design for those conditions (this is very common approach). If we do that, the controller will work at all other conditions – the performance might not be as good – but it is the best we can do with a controller with fixed parameters. We can see (Appendix A.5) that the gain K_{MISC} is directly proportional to the input voltage (V_{S0}) and we should realise that higher gains will reduce the phase and gain margin – so we should design for the maximum input voltage. Also, we can see from the transfer function $G_F(s)$ that the peak in the magnitude response and the phase shift will be increased for larger values of load resistance R – both of which will decrease the gain and phase margin – so we should design using the minimum load condition (maximum load resistance). This is maybe not so obvious, but you will be able to prove it to yourself with 5 minutes of effort using MATLAB and plotting the Bode plot for your filter for different values of load resistance.

So, step by step, the suggested design procedure for Type 1 becomes:

[1] Work out the resonant frequency of your filter and set the crossover frequency to 1/10 of this – i.e

$$\omega_C = \frac{1}{10\sqrt{LC}} \qquad (2)$$

[2] Work out the filter transfer function $G_F(s)$ for your filter (see Appendix B.1) using the maximum value of load resistance (corresponding to 25% load – see project specification).

[3] Work out the magnitude of the filter response at ω_c (it should be pretty close to unity). Use MATLAB for this – see Appendix C – it will save a lot of time if you need to do it more than once!

[4] Calculate the value of K_{MISC} – using the maximum expected value of input voltage (it is not critical to get a very accurate value – you can just use your simulation to get a sensible value). See Appendix A.5 for the derivation of K_{MISC} – you will also need to know (or have decided) what potential divider ratio you are using on the output voltage (see Appendix A.3).

[5] Work out the required value of G (the magnitude of G_C at ω_c) to get unity loop gain at ω_c. This is given by:

$$G = \frac{1}{K_{MISC}|G_F(j\omega_C)|} \qquad (3)$$

[6] Work out the value of A from: $A = G\omega_C$ (this step might seem a bit strange - why did we not just calculate A directly in step [4]? – this will become apparent if you do one of the more complex designs).

[7] Use MATLAB to check your gain and phase margins (see Appendix C). The phase margin should be fine (> 60 degrees) but you may find the gain margin is not i.e. the system will be close to instability. If it isn't, you will need to reduce ω_c and go back to [1].

[8] From the value of A, work out the values of R_1 and C_1 for the controller amplifier – see Appendix B.2.1.

Now you can try your controller in your PLECS model – it should work (PLECS has simple op-amp models and you will need to work out how to model the PWM chip). Try it with a DC input with different loads and with an AC input. Look to see if you meet the output ripple specification at 100Hz (< 0.2V) when you use an AC input – probably you won't. You can then probably decide whether to try the simple controller in hardware – or try a more complex design (below) in PLECS – or to work on both at the same time.

2.3.2 *A Better Controller (Type 2 or Type 3)*

If you design a Type 1 controller it should work fine in keeping the output voltage stable when you change the load and it should work fine if you have a DC input (from a power supply). However, if you feed your circuit from the AC input via rectifier, it will almost certainly not meet the output ripple specification at 100Hz. Remember that the output filter is totally ineffective at 100Hz – it passes straight through, so we are relying on the closed loop control to remove the 100Hz ripple. In Appendix A.6 it is shown that:

$$Output\ ripple\ \%\ at\ 100Hz = \ Input\ ripple\ at\ 100Hz/(1 + loop\ gain\ at\ 100Hz)$$

Your input ripple will be something like 10% to 20% and the output specification is 2% - so you are going to need a loop gain of something like10 or better at 100Hz to meet that.

The Type 1 controller cannot provide this – using the method in 2.2.1, the crossover frequency will be a few hundred Hz probably (depending on your filter parameters) and that is where the loop gain is unity. You can try increasing the crossover frequency, but you will not (probably) be able to make it high enough to reduce the 100Hz ripple within the specification before is becomes very oscillatory, or unstable.

The solution is to use a more complicated controller either Type 2 or Type 3. If you have time to consider a better controller, I suggest you try Type 3 – with which you can definitely meet the specification. The procedure for Type 3 is given below (if you want to try Type 2, then you can probably work it out by looking at this and looking at Venable's original paper – and using the information in Appendix B.2.2).

The transfer function of the Type 3 controller is:

$$G_C(s) = \frac{A}{s}\frac{\left(1+\frac{s}{\omega_C/\sqrt{k}}\right)^2}{\left(1+\frac{s}{\sqrt{k}\omega_c}\right)^2}$$

where *k* is Venable's k-factor (defined later).

The integrator is still there as we need that for steady state accuracy. In addition, there are two zeros at a frequency below the crossover frequency and two poles at a frequency above the crossover frequency. The effect of the zeros is to provide a phase-lead, which reduces the overall phase lag of the control loop at the crossover frequency – "fighting against" the phase lag of the output filter. The two poles have to be there, as we can never realise a transfer function with more zeros than poles – the poles give a phase lag again, but they are above the crossover frequency to minimise the effect on stability. That is a very short discussion – you can look at Venable's paper for more information – but this is pretty difficult stuff for Year 2 - so we will just concentrate now on how to implement Venable's design.

We need to choose ω_c, then we can follow Venable's method to find *k* and *G* (and hence *A*). Then we can work out the component values for the amplifier.

So, step by step, the suggested design procedure for Type 3 is:

[1] Choose a value for ω_C – with the Type 3 controller, we can put this above the filter resonant frequency – and again there is no exactly correct value. If you make it too low you won't get enough loop gain at 100Hz – if you make it too high, then you risk instability due to switching frequency noise affecting the control loop (discussion of that is way beyond the expectation of Year 2 level – so we won't discuss further). Probably putting ω_C at twice the resonant frequency will not give the performance you want and 10 times will probably lead you into trouble – so pick something in between (the design can be iterated – it is normal to do that).

[2] Work out the filter transfer function $G_F(s)$ for your filter (see Appendix B.1) using the maximum value of load resistance (corresponding to 25% load – see project specification).

[3] Work out the magnitude and phase of the of the filter response at ω_c. Use MATLAB for this – see Appendix C – it will save a lot of time if you need to do it more than once!

[4] Work out how much phase "boost" (B) you need at the crossover frequency to get a phase margin of 60^O (this is a standard design value that is often used in Bode design)

$$Phase\ Margin\ (PM) = 180 - 90 + \angle G_F(j\omega_C) + B$$

$$B = PM - 180 + 90 - \angle G_F(j\omega_C)$$

Note that in this calculation, the phase angle of $G_F(j\omega_C)$ will be negative (and remember to use degrees throughout!). You should come out with a value of $B > 70^O$ (if it is more than

150^O – you have trouble – ask for help!). The 90^O term in the equations above is due to the integrator.

[5] Venable gives a formula for calculating his k-factor in terms of the "boost" *B* (if you fancy a bit of a challenge – you can have a go at deriving it)

$$\sqrt{k} = \tan\left[{}^{B}/_{4} + 45\right] \rightarrow k = \left\{\tan\left[{}^{B}/_{4} + 45\right]\right\}^2$$

[6] Calculate the value of K_{MISC} – using the maximum expected value of input voltage (it is not critical to get a very accurate value – you can just use your simulation to get a sensible value). See Appendix A.5 for the derivation of K_{MISC} – you will also need to know (or have decided) what potential divider ratio you are using on the output voltage (see Appendix A.3).

[7] Now we calculate the required gain of the controller (*G*) at the crossover frequency:

$$G = \frac{1}{K_{MISC}|G_F(j\omega_C)|}$$

[8] Work out the value of *A* from: $A = G\omega_C/k$ (this is derived in Appendix B.2.3).

[9] Now you have all the transfer functions. Use MATLAB to check your gain and phase margins (see Appendix C). See if you have enough loop gain at 100Hz – if not try increasing ω_c and go back to [1] (for this reason, it is a good idea to put the design process into a spreadsheet (or a MATLAB script) – so you can change it easily).

[10] Now you have *G*, ω_C *and k* – that allows all the component values and (*A*) to be calculated using the results in Appendix B.2.3.

Now you can try the controller in your PLECS model – it should work. If it does, then you can try it on your hardware – it should work there as well – try it with a DC input with different loads and with an AC input. Look to see if you meet the output ripple specification at 100Hz (< 0.2V) when you use an AC input – with a Type 3 controller you should be able to.

Appendix A - System Modelling

To do control design, the things we need to model in the system are:

a) The power circuit – and in particular how the duty cycle affects the output voltage and how the ripple coming from the rectifier affects the output voltage.
b) The PWM control chip – and in particular the relationship between the duty cycle demand and the duty cycle of the pulses that are produced.
c) The amplifier circuit that we are going to use to implement the controller – and in particular how its transfer function is related to the component values.

You may think that we did at least part of (a) in EEEE2045 where we derived an expression that related the output voltage to the input voltage and the duty cycle – BUT – that was for steady state operation – now we need a model that works for transient conditions when the duty cycle and input voltage are not constant.

A.1 Power Circuit Modelling

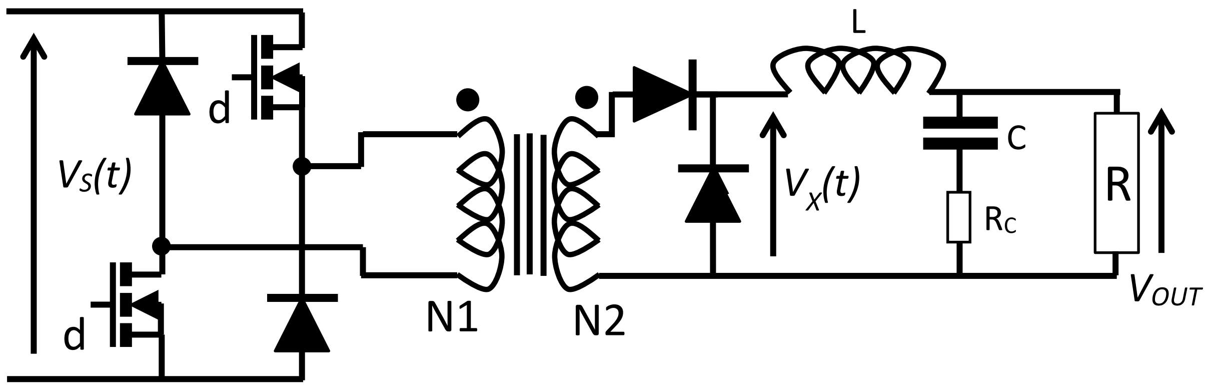
To model the power circuit, we make some assumptions:

- We model the effect of the rectifier by assuming its output is an ideal voltage source ($V_S(t)$) which has a DC (steady state) value of V_{SO} and a ripple component of $v_S(t)$.

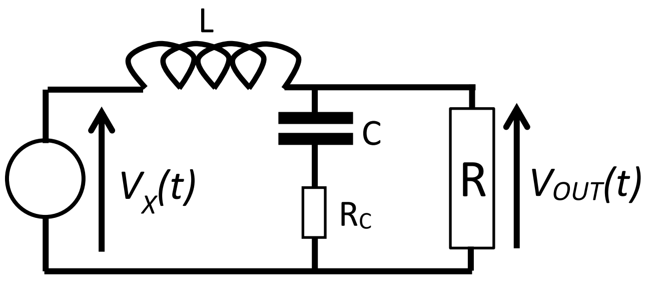
$$V_S(t) = V_{S0} + v_S(t)$$

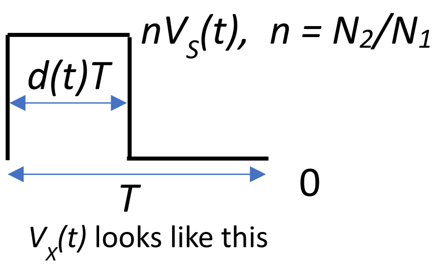
- We assume all components are ideal – except for the output capacitor where we include the "effective series resistance" – this can have an effect on the control loop. We assume the load is a resistor.

Remember the Forward Converter circuit:



In order to model this for control, we can realise that everything to the left of $V_X(t)$ simply acts like a switch to apply the voltage $V_S(t)$ (scaled by N_2/N_1) to the filter and load for a time equal to the duty cycle. We can therefore "forget" about the individual components and model their effect by a single voltage source.





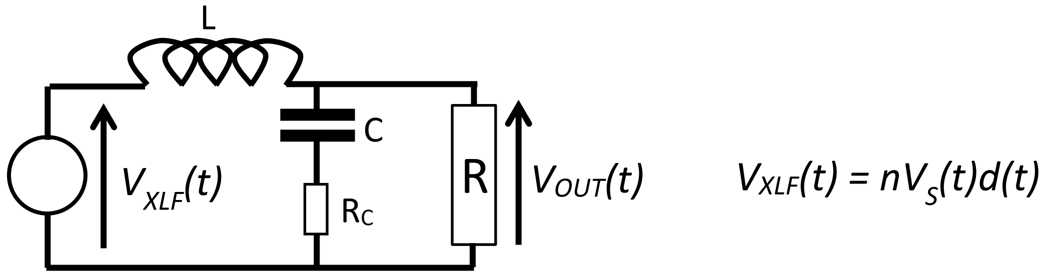
So, we made everything a lot simpler by replacing all the power devices and the transformer by a single pulsed voltage source with variable amplitude (because $V_S(t)$ has ripple) and variable duty cycle (because the controller is going to vary the duty cycle to do the control).

However, we still can't use this model very easily because it is not easy to analyse this circuit when the pulse height and pulse width are time-varying. To get around this problem – we use a "trick" that is very commonly used to develop control models of power electronic circuits – the so-called "average model approach". To do this we use the following fact:

- *If the pulse width and pulse amplitude vary at a frequency which is much lower than the switching frequency (1/T), then their effect on the output of the filter can be determined by replacing each pulse with its average value in each switching cycle.*

We won't prove this mathematically (that is for Year 4) – but hopefully you can see that it makes sense intuitively (it is easy to demonstrate with a little PLECS model). In our case the variations in $V_S(t)$ and $d(t)$ are mainly at 100Hz and the switching frequency is ≈100kHz – so this is clearly a good assumption.

Now the model becomes:



This is much better because $V_{XLF}(t)$ is now a continuous function (no pulses) – but there is still a problem with this model for control analysis. Since $V_S(t)$ and $d(t)$ appear multiplied together, this is model is non-linear. Designing control systems with non-linear models is possible – but it is very difficult and is best avoided if possible (most of the easy to use control design techniques do not work!). To get around this problem we use another "trick" – the so called "linearisation about an operating point". You will study this in Year 3/4 in more general terms – here we will just look specifically how to use it with the Forward Converter.

To make the system linear we use the idea that we can represent $V_S(t)$ as a steady state value plus a time varying value and we do the same for the duty cycle.

$$V_S(t) = V_{S0} + v_S(t)$$

$$d(t) = D_0 + \Delta d(t)$$

Here D_0 is the steady state value of the duty cycle and $\Delta d(t)$ is the variation in the duty cycle. Now we can write V_{XLF} in terms of these quantities.

$$V_{XLF}(t) = n\big(V_{S0}(D_0 + \Delta d(t)) + D_0 v_S(t) + v_S(t)\Delta d(t)\big)$$

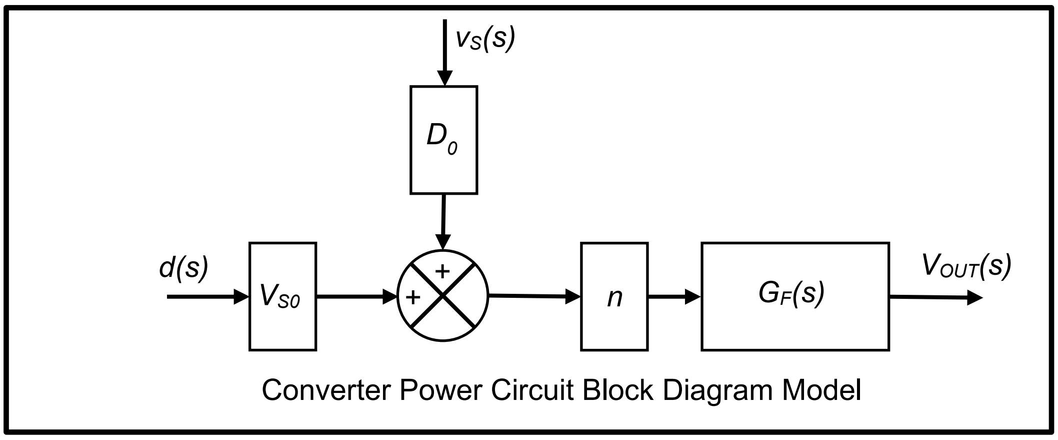
The "trick" now is to assume that the variations are small compared to the steady state values – so that the last term on the right above is small compared to the other terms and can be neglected. This is a reasonable approximation for many control systems that work around a steady state operating point (like the one we are designing here) – and its use turns what would be an extremely complex design problem into a much simpler one. $V_{XLF}(t)$ can now be written as:

$$V_{XLF}(t) = n\big(V_{S0}d(t) + D_0 v_S(t)\big)$$

This is now a linear equation that separates the effect of *d(t)* and $v_S(t)$ which is exactly what we need for the control design. The last step is to put it in a block diagram form. Defining the transfer function of the filter as $G_F(s)$ (derived later) we can write:

$$V_{OUT}(s) = G_F(s)V_{XLF}(s) = nV_{S0}G_F(s)d(s) + nD_0G_Fv_S(s)$$

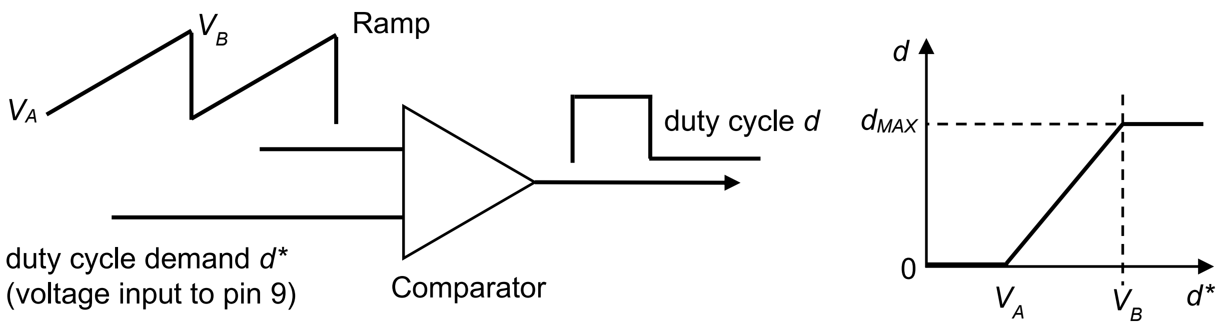
This gives us the final block diagram model for the power circuit:



Despite the slightly complicated route to get there, this is a simple model that we can use for control design. ***The model above is what you really need to know to do the design***. It is good to understand where it comes from – but it is not essential at this stage if you are struggling with it (you will meet the concepts explained in more detail in Years 3 and 4).

A.2 PWM Chip Modelling

Fortunately, modelling the PWM chip is quite easy. The relevant section inside the 3424 chip looks like (this is simplified and does not include the output stage etc – but has enough for what we need):



The graph shows how the actual duty cycle (*d*) will vary with the duty cycle demand (*d**). The important part for the control loop is the slope of the line in the active part which defines the "gain" (K_{PWM}) of the PWM generation and is given by:

$$\boxed{K_{PWM} = \frac{d_{MAX}}{V_B - V_A}}$$

V_A and V_B can be found from the data sheet and d_{MAX} is typically 45% (0.45).

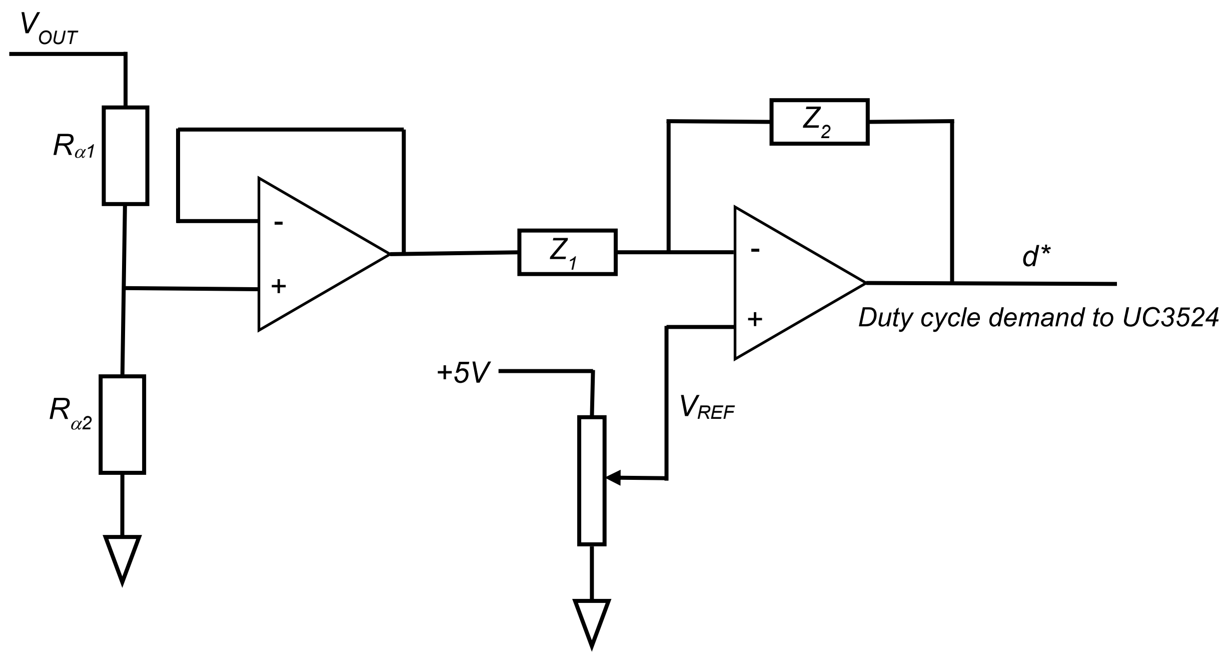
You may be worried about the “offset” caused by V_A (i.e the fact that the duty cycle demand has to reach V_A before the any pulses at all are generated). In fact, that has no effect on the control loop analysis because the controller is going to have an integrator – which will automatically compensate for any offsets. In control block diagram, the PWM chip can simply be represented by:

$$\begin{tikzcd} {} \arrow[r, "d^*(s)"] & \boxed{K_{PWM}} \arrow[r, "d(s)"] & {} \end{tikzcd}$$

PWM Chip Block Diagram Model

A.3 Amplifier (controller) Modelling

The amplifier circuit that you are going to use to implement the controller looks like this (power supplies etc omitted):

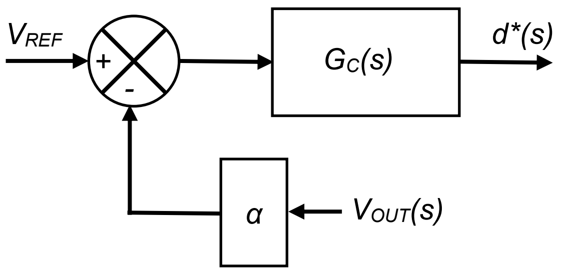


V_{OUT} is the output voltage of the converter, the potentiometer allows you to adjust V_{REF} and the potential divider ($R_{\alpha 1}$, $R_{\alpha 2}$) allows the output voltage to be scaled down to a suitable value. You will need to think about what a suitable scaling might be – depending on the voltage the op-amps are supplied with, and the value of V_{REF} you decide to set. Z_1 and Z_2 set the frequency response of the amplifier – they are the main parts that need to be designed (more detail below). You should be able to satisfy yourself that the output of this circuit is related to the input by:

$$d^*(s) = \left(V_{REF} - \alpha V_{OUT}(s)\right)G_C(s) + V_{REF}$$

$$G_C(s) = \frac{Z_2(s)}{Z_1(s)}, \qquad \alpha = \frac{R_{\alpha 2}}{R_{\alpha 1} + R_{\alpha 2}}$$

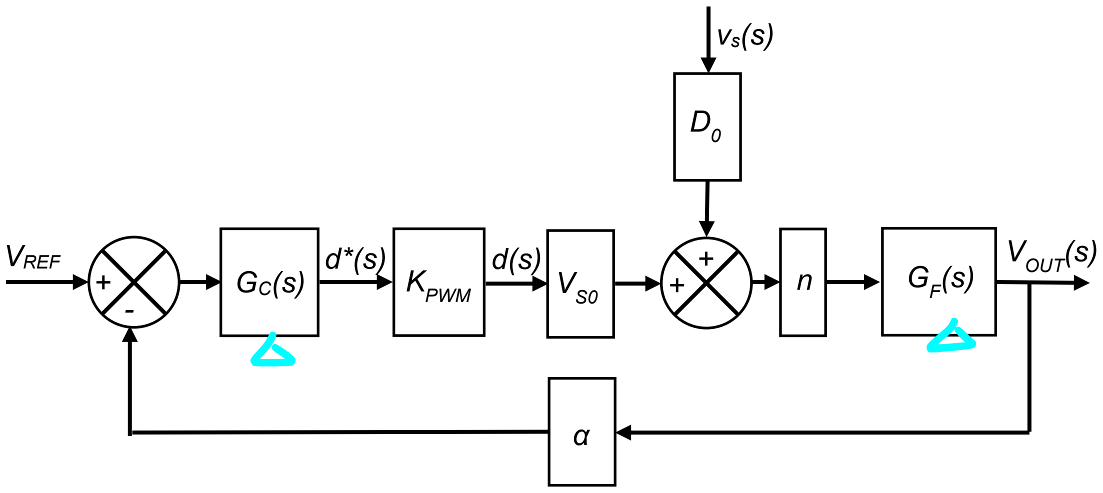
G$G_C(s)$ is the transfer function of the amplifier (discussed in more detail later) and α is the potential divider ratio. The first term in the equation for d^* is just the error between the scaled output voltage and the reference. The second term is just a DC offset in the amplifier output – it will make no difference to the control loop analysis because the integrator that will form part of G_C will compensate for it and we can ignore it (just like we did for the offset in the PWM chip). The amplifier block diagram is therefore:



Amplifier Circuit Block Diagram Model

A.4 Complete Block Diagram Model

We now have all the individual models and can assemble the complete block diagram below.

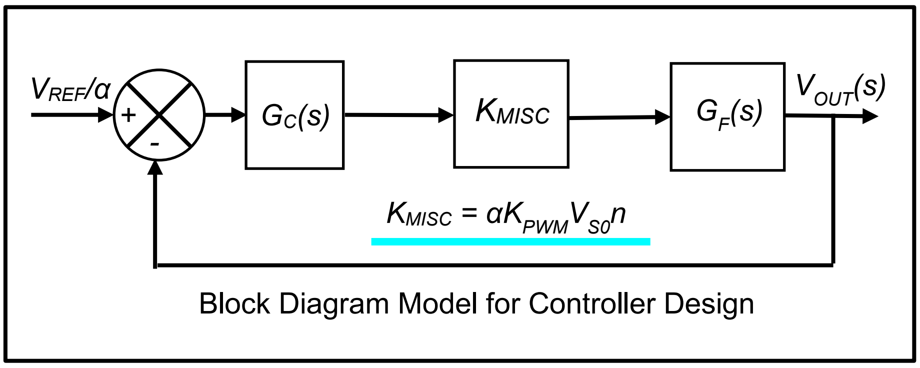


Complete Block Diagram Model

This model will enable us to do the control design for $G_C(s)$ and it will also enable us to determine the influence of $v_S(s)$ (the ripple in the rectifier voltage) on the output voltage for any design of $G_C(s)$. It is useful to re-arrange the block diagram to look at these in more detail.

A.5 Block Diagram Model for Controller design

To design the controller $G_C(s)$ (to work out phase margin etc, closed loop gain etc) we just need to consider the main loop and can ignore the $v_S(s)$ (i.e. we can set it to zero). Also, it is useful to re-arrange the diagram to get unity gain feedback (using the techniques you have studied in EEEE2045). The diagram becomes:



Block Diagram Model for Controller Design

Here the effect of the potential divider, the PWM chip, the steady state input voltage and the transformer turns ratio have all been combined into one block with a single gain K_{MISC}.

A.6 Block Diagram Model to determine 100Hz ripple in V_{OUT}

This is a bit more complicated – the final result is the important one. To work out the influence of the ripple in V_S on the output voltage V_{OUT} we again consider that V_S can be split into 2 parts (a steady state DC part and the ripple part):

$$V_S(t) = V_{S0} + v_S(t)$$

Now we do the same with the output voltage V_{OUT}:

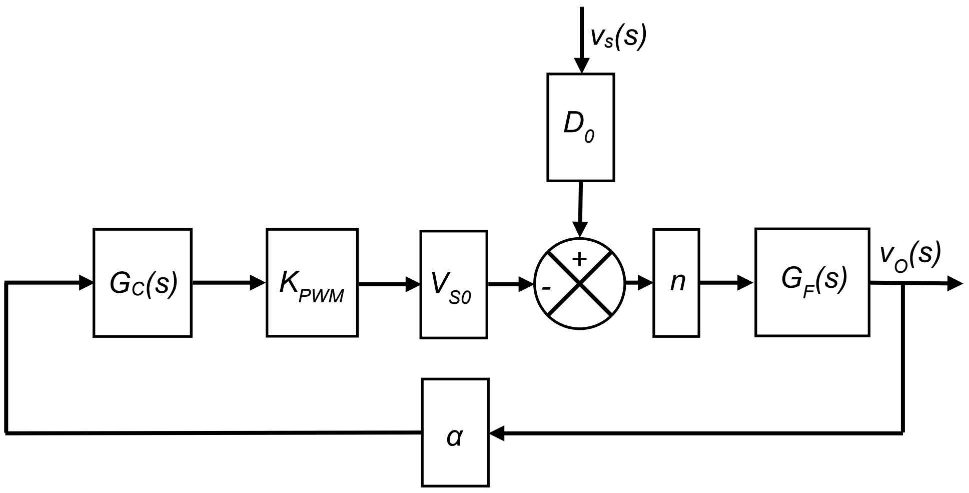
$$V_{OUT}(t) = V_{DC} + v_O(t)$$

V_{DC} is just the DC output (10V in your case) while $v_O(t)$ is the ripple in the output created by the ripple at the input ($v_S(t)$). We want to know how the % ripple at the output ($v_O(t)/V_{DC}$) is related to the % ripple at the input ($v_S(t)/V_{SO}$) and the control loop design.

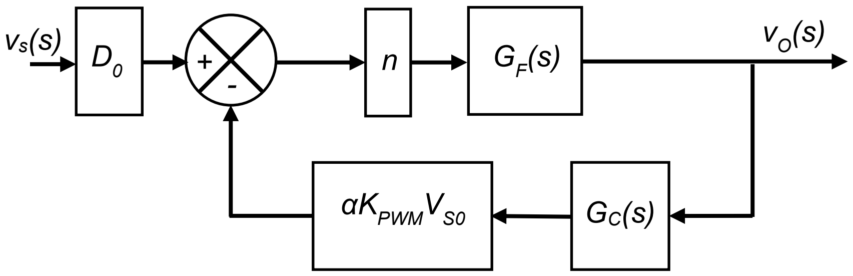
The first step to relate the DC values is easy:

$$V_{DC} = nD_O V_{SO}$$

where D_O is the steady state duty cycle. To determine $v_O(t)$ in terms of $v_S(t)$ we can re-arrange the block diagram and use superposition (i.e. we can ignore V_{REF} in the diagram if we are just interested in the effect of $v_S(t)$). The diagram becomes:



Which can be re-arranged into a more familiar form:



Applying what you learnt in EEEE2045 about block diagrams – you should be able to satisfy yourself that $v_O(s)$ is related to $v_S(s)$ by:

$$\frac{v_O(s)}{v_S(s)} = \frac{nD_O G_F(s)}{1 + n\alpha K_{PWM} V_{SO} G_F(s) G_C(s)}$$

Therefore (using the steady state relationship from before) we can see that:

$$\frac{v_O(s)}{V_{DC}} = \left(\frac{v_s(s)}{V_{SO}}\right) \frac{G_F(s)}{1 + n\alpha K_{PWM} V_{SO} G_F(s) G_C(s)}$$

To work out the effect at 100Hz we need to we need to replace s by $j\omega$ and evaluate at ω corresponding to 100Hz. That looks a bit complex – but in fact it is quite easy if we realise that the gain of the filter G_F at 100Hz is practically unity and that the complicated looking expression in the denominator is just the total loop gain plus 1. Therefore, we can get the very simple result: ?

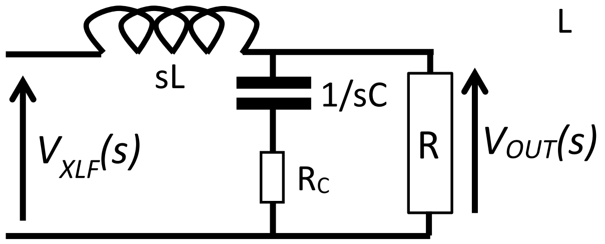
$$\boxed{Output\ ripple\ \%\ at\ 100Hz = \ Input\ ripple\ at\ 100Hz/(1 + loop\ gain\ at\ 100Hz)}$$

So, for example, if the input ripple is 10% and we want 1% at the output, then the loop gain at 100Hz will have to be at least 9. When you design your controller – you can use this to work out if it will enable your power supply to meet the ripple specification.

Appendix B – Transfer Functions

To do the control design, we need to be able to calculate the transfer functions *$G_F(s)$* and *$G_C(s)$.* These are considered below in detail.

B.1 Output Filter Transfer Function



To get the transfer function *$G_F(s) = V_{OUT}(s)/V_{XLF}(s)$,* we represent the inductor and capacitor in the Laplace domain and perform a standard circuit analysis (the details are not give here – you should be able to do it) which gives:

$$G_F(s) = \left(\frac{R}{R + R_C}\right)\frac{(1 + sR_CC)}{\left(S^2LC + S\left(\frac{L + CRR_C}{R + R_C}\right) + \frac{R}{R + R_C}\right)}$$

For any sensible design and operating condition *$R \gg R_C$* and this can therefore be simplified to:

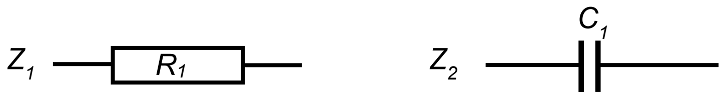
$$\boxed{G_F(s) = \frac{(1 + sR_CC)}{\left(S^2LC + S\left(\frac{L}{R} + CR_C\right) + 1\right)}}$$

B.2 Amplifier Transfer Functions

Assuming the control design we assume that the amplifier is one of three types (Type 1, Type 2 or Type 3) depending on the configuration of *Z_1* and *Z_2*. These are considered in more detail below. The important results for your control implementation are the ones that allow the component values to be calculated.

B.2.1 Type 1 Amplifier

This is just an integrator with the following configurations for *Z_1* and *Z_2*.



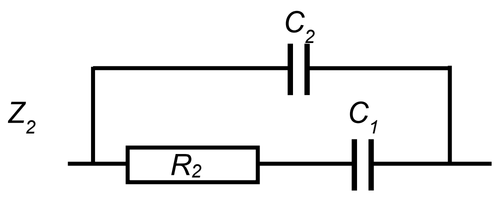
The transfer function and component values are:

$$\boxed{G_C(s) = \frac{A}{s} \qquad A = \frac{1}{C_1 R_1}}$$

Note that R_1 is not unique and has to be chosen first (a sensible choice is needed to make sure the capacitor value is sensible).

B.2.2 Type 2 Amplifier

This has a more complex configuration for Z_2 (Z_1 is the same as Type 1).



The transfer function is given by:

$$G_C(s) = \frac{A\,(1 + s/\omega_Z)}{s\,(1 + s/\omega_P)}, \qquad A = \frac{1}{(C_1 + C_2)R_1}, \quad \omega_Z = \frac{1}{C_1 R_2}, \quad \omega_P = \frac{(C_1 + C_2)}{C_2 C_1 R_2}$$

In order the relate this to the control design method of Venable we need to consider his definitions:

> ω_C = crossover frequency (the frequency at which the total loop gain is 1)
> G = the gain of the amplifier at the crossover frequency
> k = Venable's k-factor which defines the pole and zero frequencies in terms of the crossover frequency $\omega_Z = \omega_C / k\,, \omega_p = k\omega_C$ (for Type 2).

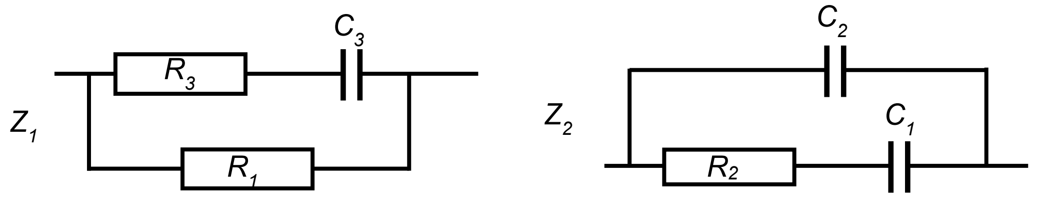
We can now work out the component values in terms of Venable's design parameters

$$\boxed{C_2 = \frac{1}{kR_1\omega_C G}, \quad C_1 = C_2(k^2 - 1), \quad R_2 = \frac{k}{\omega_C C_1}, \qquad G = \frac{Ak}{\omega_C}}$$

Note that R_1 is not unique and has to be chosen first (a sensible choice is needed to make sure the other component values are sensible – some iteration may be required).

B.2.3 Type 3 Amplifier

This has a more complex configuration for Z_1 (Z_2 is the same as Type 2).



The transfer function is given by:

$$G_C(s) = \frac{A}{s}\frac{\left(1+\frac{s}{\omega_{Z1}}\right)\left(1+\frac{s}{\omega_{Z2}}\right)}{\left(1+\frac{s}{\omega_{P1}}\right)\left(1+\frac{s}{\omega_{P2}}\right)}$$

$$A = \frac{1}{(C_1+C_2)R_1}, \quad \omega_{Z1} = \frac{1}{C_1R_2}, \quad \omega_{P2} = \frac{(C_1+C_2)}{C_2C_1R_2}, \omega_{Z2} = \frac{1}{C_3(R_1+R_3)}, \quad \omega_{P2} = \frac{1}{C_3R_3}$$

In order the relate this to the control design method of Venable we need to consider his definitions:

ω_C = crossover frequency (the frequency at which the total loop gain is 1)
G = the gain of the amplifier at the crossover frequency
k = Venable's k-factor which defines the pole and zero frequencies in terms of the crossover frequency $\omega_Z = \omega_C/\sqrt{k}\,, \omega_p = \sqrt{k}\omega_C$ (for Type 3).
Also $\omega_{Z2} = \omega_{Z1}, \ \omega_{p2} = \omega_{p1}$ for Type 3

We can now work out the component values in terms of Venable's design parameters

$$\boxed{C_2 = \frac{1}{R_1\omega_C G}, \quad C_1 = C_2(k-1), \quad R_2 = \frac{\sqrt{k}}{\omega_C C_1}, \quad R_3 = \frac{R_1}{(k-1)}, \quad C_3 = \frac{1}{R_3\omega_C\sqrt{k}}, \quad G = \frac{Ak}{\omega_C}}$$

Note that R_1 is not unique and has to be chosen first (a sensible choice is needed to make sure the other component values are sensible – some iteration may be required).

Appendix C – Some useful hints on using MATLAB

Transfer functions can be input directly into Matlab using the 'tf' (transfer function) command, with an array which represents the numerator coefficients, and one for the denominator coefficients. For example:

$$Gp(s) = \frac{(5682)}{s^2 + 0.5682s + 5682}$$

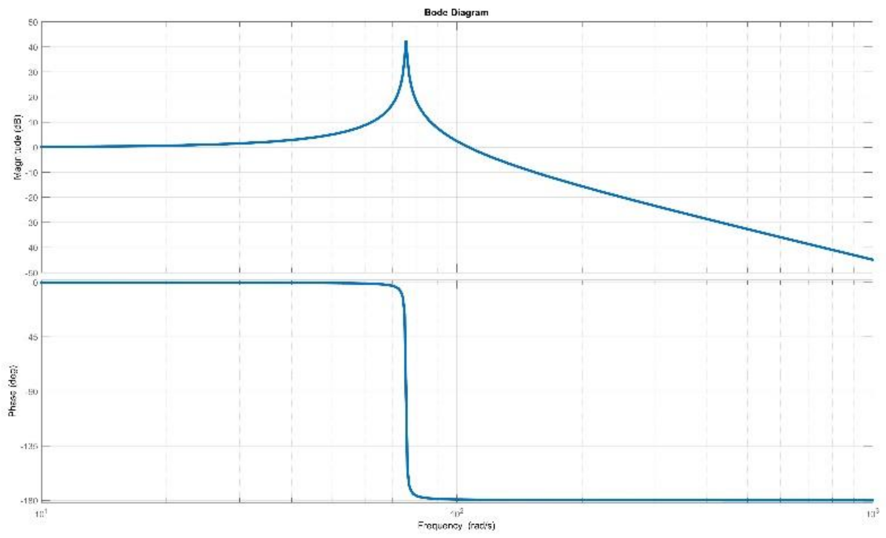
The array for the numerator, (5682) is [5682]
The array for the denominator (s^2+0.5682s+5682) is [1 0.5682 5682].

This can be input into Matlab in the command window by typing (note Gp is an arbitrary name):

Gp= tf([5682], [1 0.5682 5682])

Once you have input the transfer function for your system a Bode plot can be plotted simply with the command: bode(Gp);

This gives the response in this case of that in figure 1.



Bode Plot of function plotted in Matlab

There are also various other commands in MATLAB that can be useful for Control including *step(Gp)* [i.e. the step response], *impulse(Gp)* [i.e. the impulse response] etc.

Some particularly useful commands for this design exercise are:

margin(Gloop) – this plots a Bode plot and calculates the gain and phase margins if Gloop is the open loop transfer function.

Note that in order to get Gloop – you can multiply transfer functions together in the same way that you would with variables in MATLAB – for example using:

```
Gloop = Gcontroller*Gplant
```

at the command line, where Gcontroller and Gplant have both been defined using the tf command.

You can also calculate the closed loop transfer function using something like:

```
Gclosedloop = Gloop/(1 + Gloop)
```

at the command line, then you can use the “step” command to get the step response, or you can use the Bode command to get the closed loop Bode plot (to look at the bandwidth for example).

It is a very powerful tool! – experiment!

Other useful information

[1] Al Watson, “Bode Plots in Software”, November 2017
Information on how to plot Bode plots in various software packages including PLECs, MATLAB and LTSpice.

[2] Venable, H. Dean. “The K Factor: A New Mathematical Tool for Stability Analysis and Synthesis.” Proc. Powercon 10. 1983. San Diego, CA. pp. H1−1 to H1−12
Venable’s original paper on the k-factor Method- Note, this is a difficult- the method has been “tweaked” in this handout

All of these references are in the “Extra Resources for Control Design” on the EEEE2046 Moodle site.

Jon Clare and Al Watson, Updated January 2020