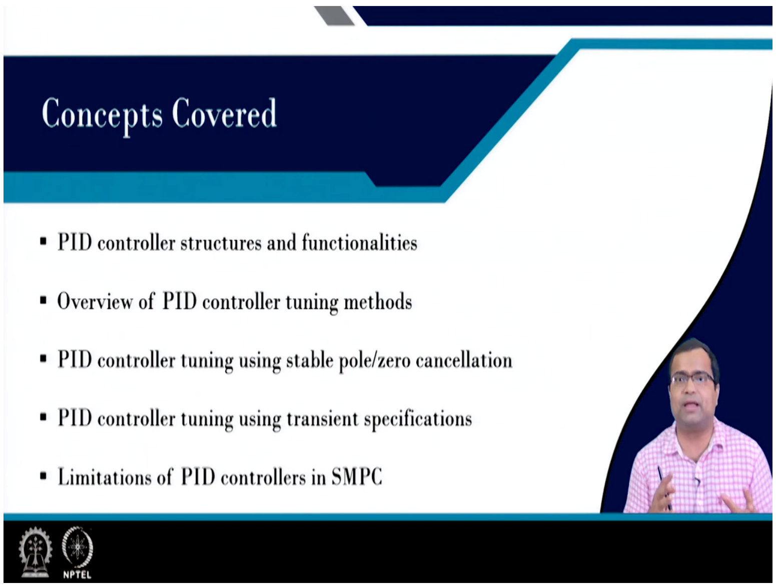
# Control and Tuning Methods in Switched Mode Power Converters

**Prof. Santanu Kapat**
**Department of Electrical Engineering**
**Indian Institute of Technology, Kharagpur**

## Module - 07
## Small-signal Design and Tuning of PWM Voltage Mode Control
## Lecture - 34
## PID Control Design and Tuning Under VMC with MATLAB Case Studies

Welcome this is lecture number 34. In this lecture we are going to talk about PID Controller Design and Tuning Under Voltage Mode Control with MATLAB Case Studies.

(Refer Slide Time: 00:36)

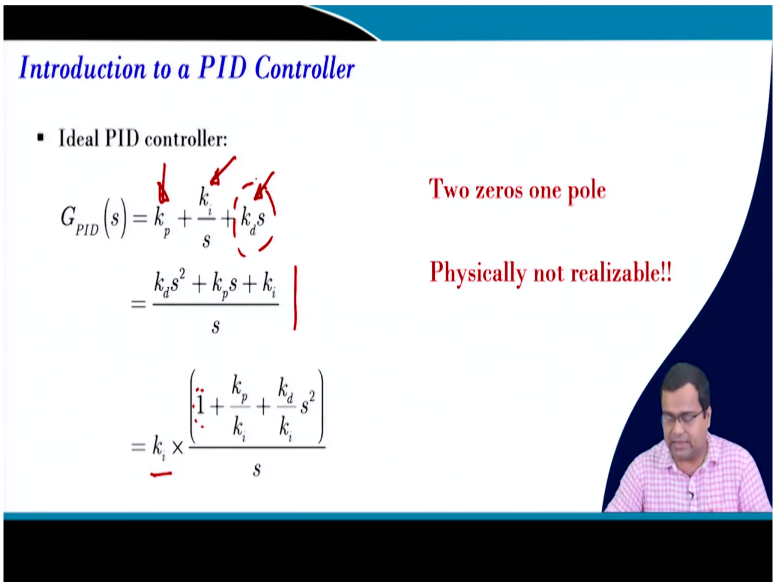


### Concepts Covered

- PID controller structures and functionalities
- Overview of PID controller tuning methods
- PID controller tuning using stable pole/zero cancellation
- PID controller tuning using transient specifications
- Limitations of PID controllers in SMPC

So, here we are going to talk about first PID controller structures and functionality, then overview of PID controller tuning methods, then PID controller tuning using stable pole 0 cancellation, then PID controller tuning using, transient specifications, and the limitation of PID controller in switch mode power converter.

(Refer Slide Time: 00:58)



## Introduction to a PID Controller

- Ideal PID controller:

$$G_{PID}(s) = k_p + \frac{k_i}{s} + k_d s$$

$$= \frac{k_d s^2 + k_p s + k_i}{s}$$

$$= k_i \times \frac{\left(1 + \frac{k_p}{k_i} + \frac{k_d}{k_i} s^2\right)}{s}$$

Two zeros one pole

Physically not realizable!!

So, first we want to talk about a PID controller, which is a very well known controller in the context of the control system. In the PID controller, if you take an ideal PID controller, it is a parallel realization. So, we have a proportional gain, which is k p proportional gain, an integral gain which is k i and a derivative gain which is k d ok. This is a standard PID controller.

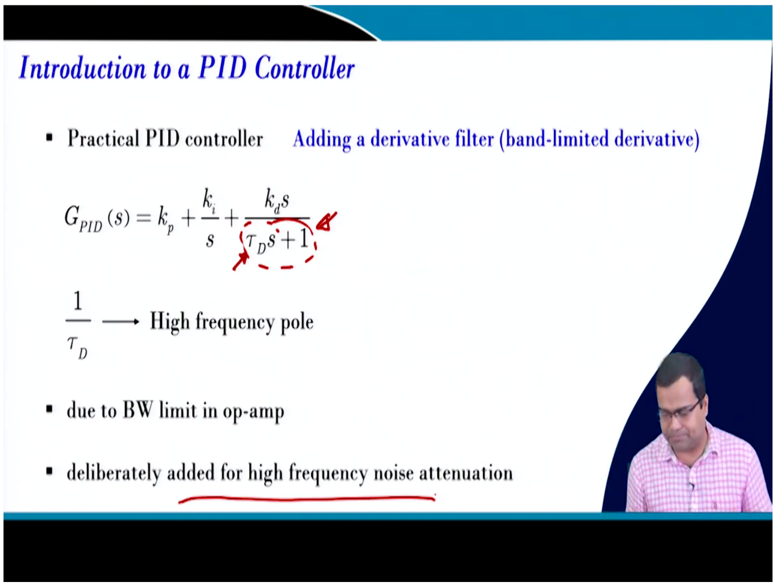
And, if we express this PID controller into polynomial numerator and denominator polynomial form. So, this is the structure. And, then we can separate out k i, because we want to write in this form where the coefficient in the numerator we want to keep you know without; that means, the constant coefficient we have to write in terms of unity, ok.

Because, you know, for power converter in the previous lecture, what we did that we all the time numerator and denominator we wrote in this form where the constant term was written was written in terms of unity by suitably scaling it. Next, this PID controller has two zero one pole, but this is not physically realizable. Because we know that this is an improper transfer function, we need at least two pole two zero means there should be at least two poles ok.

Now, introduction and we will discuss this in actual you know PID controller reality. We cannot implement an ideal derivative. So, there will be always a band limited derivative where we need to consider a derivative filter; that means a low-pass filter.

And, this comes typically when you use an op amp; the op-amp itself has a finite bandwidth that is coming into the picture that comes into the picture.

(Refer Slide Time: 02:43)



## Introduction to a PID Controller

- Practical PID controller   Adding a derivative filter (band-limited derivative)

$$G_{PID}(s) = k_p + \frac{k_i}{s} + \frac{k_d s}{\tau_D s + 1}$$

$$\frac{1}{\tau_D} \longrightarrow \text{High frequency pole}$$

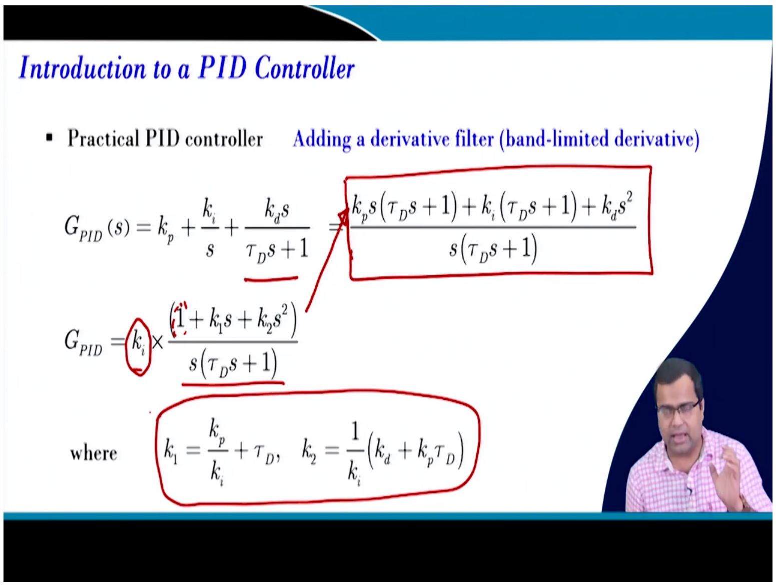
- due to BW limit in op-amp
- deliberately added for high frequency noise attenuation

Now, introduction the practical PID controller, we need to add a derivative filter. As we discuss and this derivative filter comes in this structure, where tau D is the time constant and typically this time constant is much faster. If you want to realize a derivative action, depending upon the type of system we are considering, because this should be much smaller than the time constant of the system.

Then, 1 by tau D is a high frequency pole that we are introducing and typically due to band limited emit of the op-amp, this by default comes. But, even we can set this pole much lower than the actual op-amp bandwidth. So; that means, this can be even lower bandwidth than the bandwidth of the op-amp, because we even can see the if the op-amp has 200 megahertz bandwidth and your power converter is 1 megahertz. Then, 200 megahertz is too high. So, this derivative action can inject various high frequency noise into the system and that is not desirable.

Sometime you deliberately add a high frequency pole for noise attenuation that I told; that means, it can be much smaller than the actual bandwidth of the op-amp.

(Refer Slide Time: 04:00)



## Introduction to a PID Controller

- Practical PID controller    Adding a derivative filter (band-limited derivative)

$$G_{PID}(s) = k_p + \frac{k_i}{s} + \underline{\frac{k_d s}{\tau_D s + 1}} = \boxed{\frac{k_p s(\tau_D s+1) + k_i(\tau_D s+1) + k_d s^2}{s(\tau_D s+1)}}$$

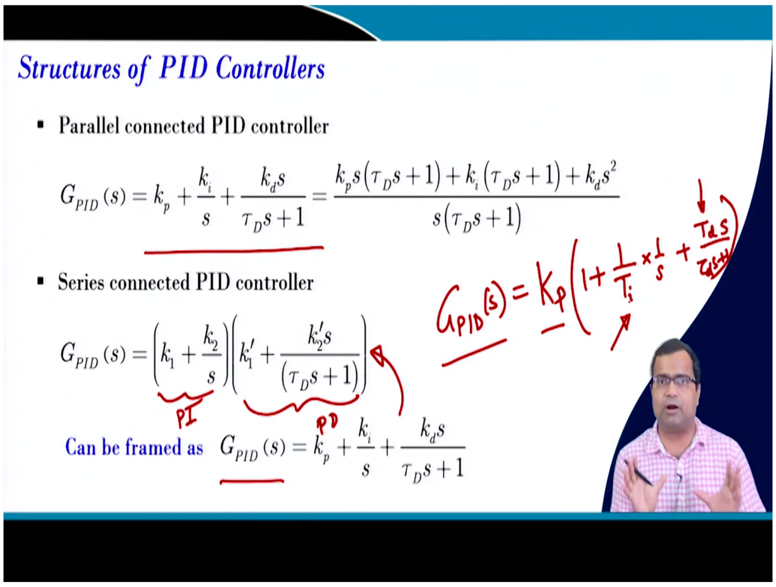
$$G_{PID} = k_i \times \frac{(1 + k_1 s + k_2 s^2)}{\underline{s(\tau_D s+1)}}$$

where $k_1 = \frac{k_p}{k_i} + \tau_D, \quad k_2 = \frac{1}{k_i}(k_d + k_p \tau_D)$

So, now, a PID practical PID controller, I think this can be written as with a derivative filter in this structure; that means the pole 0 structure in numerator and denominator coefficient form. Where again we can write our standard notation; that means, all the time we try to write the constant term should be unity by suitably scaling. So, here we took out k i.

So, whether it is an ideal PID controller or a practical PID controller. This DC gain of this PID controller always is your; that means, your k i. In this case, but yes even integrator, I am saying that the constant term which is taking out is the k i. So, where we can write that how can this can be mapped with this coefficient and that will lead to this mapping between k 1, k 2 with the k p k i and k d and tau D.

(Refer Slide Time: 04:55)



## Structures of PID Controllers

- Parallel connected PID controller

$$G_{PID}(s) = \underline{k_p + \frac{k_i}{s} + \frac{k_d s}{\tau_D s+1}} = \frac{k_p s(\tau_D s+1) + k_i(\tau_D s+1) + k_d s^2}{s(\tau_D s+1)}$$

- Series connected PID controller

$$G_{PID}(s) = \underbrace{\left(k_1 + \frac{k_2}{s}\right)}_{PI}\underbrace{\left(k_1' + \frac{k_2' s}{(\tau_D s+1)}\right)}_{PD}$$

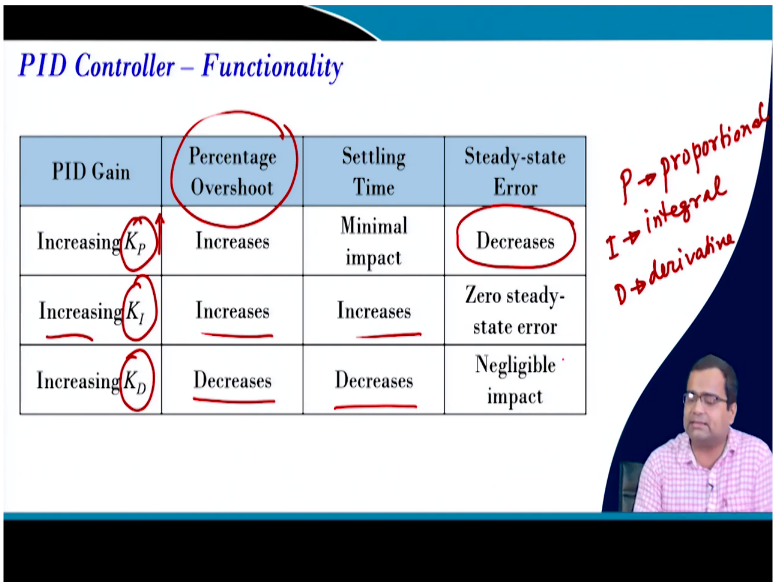
Can be framed as $\underline{G_{PID}(s)} = k_p + \frac{k_i}{s} + \frac{k_d s}{\tau_D s+1}$

$$\underline{G_{PID}(s)} = K_p\left(1 + \frac{1}{T_i}\times\frac{1}{s} + \frac{T_d s}{\tau_d s+1}\right)$$

Now, structure of PID controller; so, PID controller, whatever we discuss is a parallel PID controller. And, of course, this is with a derivative filter, but we can also consider a series PID controller. In which we can consider a simply PI control and a PD control in cascade and that can be transformed into a PID controller.

And, where this coefficient can be mapped and it can be obtained k p k i from this k 1 k 2 k 1 dash k 2 dash. So, it can be a function of this; that means, k p k i k d will be a function of k 1 k 1 dash k 2 k 2 dash and it can be function of tau D.

(Refer Slide Time: 05:43)



## PID Controller – Functionality

| PID Gain | Percentage Overshoot | Settling Time | Steady-state Error |
|---|---|---|---|
| Increasing $K_P$ | Increases | Minimal impact | Decreases |
| Increasing $K_I$ | Increases | Increases | Zero steady-state error |
| Increasing $K_D$ | Decreases | Decreases | Negligible impact |

P → proportional
I → integral
D → derivative

So, we talked about parallel PID controller and series PID controller. Sometime PID controller is also written like this, PID in the parallel form it is also sometime we can take k p out, then we write 1 plus you know 1 by T i the time constant into 1 by S that is the integral term plus T d which is a derivative term into s.

So, this also is another form and, of course, there is the derivative filter; it is there. So, we can take k p out and then we can write in terms of the integral time constant and the derivative time constant. This is also another way of representation, where k p is common ok. And, this type of structure is used, you know, particularly for Ziegler Nichols tuning, where the k p can be obtained from you know the critical gain.

And, then the timing parameter like integral timing, this integral time as well as derivative time constant this those can be found from their suitable tuning rule. So, PID controller functionality; PID controller it has three actions – one is the proportional, which links with the proportional term proportional, then I stands for integral it stands for integral and D stands for derivative and their respective gains are K P, K I and K D.

Now, if we do not change K I and K D. If we increase K P; that means, if we increase K P, then overshoot will increase percentage overshoot. Because, we will see that if you draw the Bode plot frequency response. If you simply increase the proportional gain in most of the cases that your gain plot will simply go up and as a result, your bandwidth can increase.

But this can also reduce the phase margin, because your phase response may not drastically change and that can reduce the phase margin. As a result, it will lead to a poor phase margin and that increases the percentage overshoot. It has reduced I mean very negligible impact on the settling time, but of course, it has impact in the rise time. That means a higher proportional gain tries to reduce the rise time, but at the same time, it increases the overshoot undershoot.

And, of course, higher proportional gain can reduce the steady state error, but it cannot achieve 0 steady state error ok. Whereas, the integral action is used to reduce the steady state error, and in if you wait for a very long time like limit T tends to infinity, then you can get 0 steady state error provided that if the input is a step input. But, integral action cannot achieve 0 steady state error if the input is sinusoidal or something else, and that comes from the theory of internal model control.

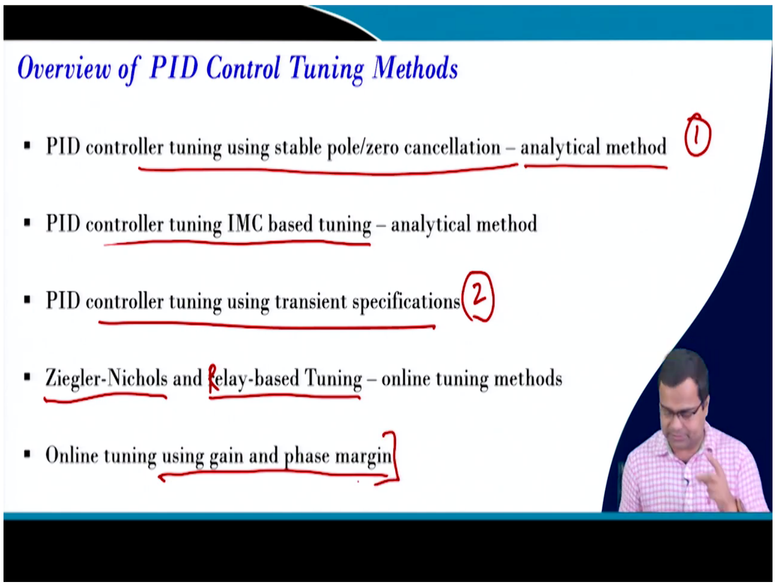
Now, for the integral control because most of the cases that we are going to consider is a step input. So, PID controller; that means, integral action will inherently try to achieve very reduced steady state error or almost negligible steady state error. If we increase the integral gain, then the settling time will increase. Because, it will be more oscillatory overshoot will increase, that is because it will reduce the phase margin ok.

At the same time, it will also increase the settling time, but in some cases you know when we discuss current mode control, because as well as the voltage mode control. So, we discuss feed forward action. So, if you do not consider feed forward action, you know in voltage mode control we will see the loop gain depends on the input voltage.

So, in order to; that means, the loop gain can vary when there is a change in input voltage; that means, if at lower input voltage, we will see the lower loop gain. In order to anticipate we need to increase that integral gain. Because we need to achieve certain DC gain and this higher integral gain can be problematic, because it can cause you know higher, you know poor phase margin, but you know in analog control because we design all this PID controller coefficient offline.

So, it design at the very beginning. But, in digital control you can tune it, but this integral action if we increase to in reduce that is you know a steady state error, it has an impact on the transient response. Now, the derivative control is used, which is trying to provide more damping. So, that it reduces the overshoot, because it tries to increase the phase margin. And, it also try to reduce the settling time because it tries to damp the system right, but it has a negligible impact on the steady state error ok.

(Refer Slide Time: 10:57)



## Overview of PID Control Tuning Methods

- PID controller tuning using stable pole/zero cancellation – analytical method ①
- PID controller tuning IMC based tuning – analytical method
- PID controller tuning using transient specifications ②
- Ziegler-Nichols and Relay-based Tuning – online tuning methods
- Online tuning using gain and phase margin

Next, we want to discuss, what are the different PID controller tuning rule? So, one of the rule tuning rule that we are going to consider in this particular lecture is PID controller tuning using stable pole zero cancellation, and that is something analytical method, like this is an offline method.

The second one PID controller tuning using Internal Model Control IMC based tuning, I am not going to discuss, but you can refer to standard textbook, where these techniques are discussed in elaborate you know in detail. Another specification of PID controller tuning using transient specification. We are going to consider in this particular using MATLAB case study. So, we will consider this as well as this MATLAB case study, where using we will use a MATLAB toolbox for this particular 2nd case.

And, there we can specify the transient time and then, based on some optimization criteria, the PID controller can be obtained the parameter. And, then accordingly those parameters can be imported and we can use that controller and we want to see the response.

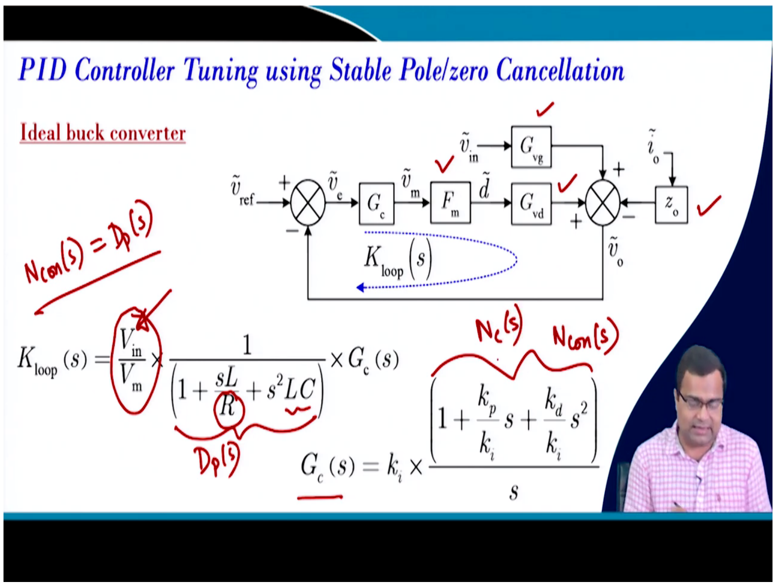
Online tuning method the Ziegler Nichols tuning method is a popular method, but the difficulty in this method. This method requires in order to find the k p k i k d; that means the controller gain. You need to run the system under critical condition where it will be oscillatory. And, that is something is not desirable in a power converter, because it may lead to the complete collapse of the system or it can damage the devices.

In order to solve this problem, the relay based tuning was proposed. You know this is a very effective technique where we can generate this oscillation by means of a relay feedback ok. And, we will discuss this technique the methodology in the subsequent in this lecture ok.

Where, we can use either a on off non-linearity or a hysteresis non-linearity to forcefully create oscillations, but control oscillation. From that oscillation, we can measure the amplitude of oscillation as well as the time period and those can be used to get the tuning parameters.

Then, there are other method of tuning online tuning using frequency response method where, if you set some you know gain margin and phase margin desired gain margin and phase margin you can actually tune the parameter ok. So, this is also online tuning method. And, there are multiple papers because we are not talking about digital control. You know multiple pioneer works actually are already proposed in this context.

(Refer Slide Time: 13:39)



## PID Controller Tuning using Stable Pole/zero Cancellation

Ideal buck converter

$N_{con}(s) = D_p(s)$

$$K_{loop}(s) = \frac{V_{in}}{V_m} \times \underbrace{\frac{1}{\left(1+\frac{sL}{R}+s^2LC\right)}}_{D_p(s)} \times G_c(s)$$

$N_c(s)$ $N_{con}(s)$

$$G_c(s) = k_i \times \frac{\overbrace{\left(1+\frac{k_p}{k_i}s+\frac{k_d}{k_i}s^2\right)}^{N_c(s)}}{s}$$

So, now we are going to first talk about PID controller using tuning using stable pole zero cancellation. So, we want to start with an ideal buck converter and this is the loop gain plot. In fact, this we have discussed in the previous lecture, that how to obtain this loop gain right?

Because we already obtained, you know the modulator gain; we obtain already the control to output transfer function audio susceptibility output impedance. In fact, we have chain we have verified this model open loop model, using time domain simulation; that means, we have checked with the response of the linearized system, and the response of the original switch system. And, we have verified that ok.

That means, we have verified the models are reasonably accurate. At least you know within the control bandwidth, but we have not talked about what is the control bandwidth? How much is the control bandwidth that we can achieve ok? So, ideal buck converter this is a loop transfer function and then if we take a PID controller, ideal PID controller, we are going to talk about the realization aspect.

But, then what is our objective our initial objective we need to cancel; that means, this the numerator of the controller that polynomial, we need to set to the denominator of this plan ok.

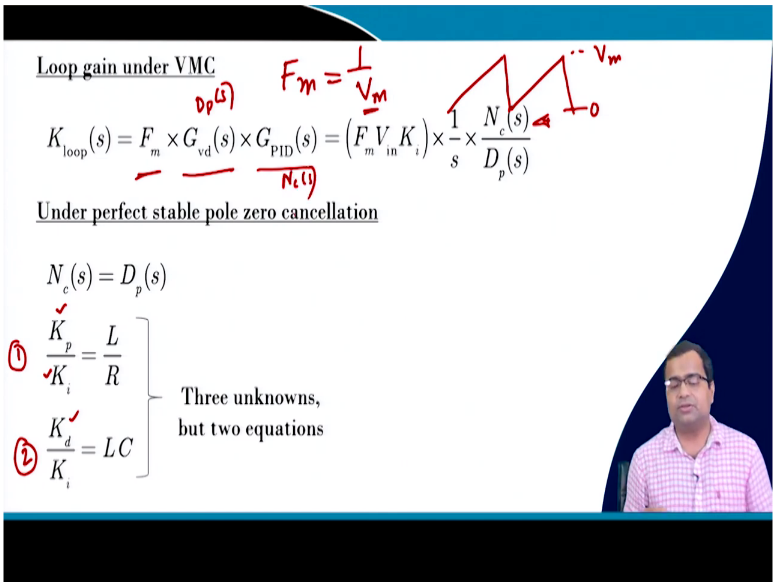
That means, what we want the numerator of this controller S, we want to make exactly equal. But, I will discuss you know if this is a non-robust compensation and in the next lecture we will discuss, if you cannot exactly cancel, then what could be a better way. Because, the problem in this method, LC parameters are reasonably known, because they are known from the design, because you have selected L and C though there can be slight variation like a 10 percent variation plus minus LC.

But, resistance is something which is which varies. So, you do not know what will be the resistance value right. So, it is, and it is not easy to know, because you know you can use a load estimator, but that is not a good technique. So; that means, we need a robust compensation where the load resistance will not come in the compensation process.

Another part is the loop transfer function you see. It depends on two things. One is the input voltage, which we told that is the drawback of this voltage mode control, because the loop gain is dependent on input voltage. And, we will show in subsequent lecture when you will compare, this dependency of the input voltage in a loop gain can be overcome, by considering an input voltage feed forward ok.

So, by using an input voltage because we saw in lecture number 14 as well as lecture number I think it was in 17, where we have discussed the feed forward control. Along with the feedback, where we can achieve the input voltage insensitive, like an input voltage invariance in the loop gain by means of feed forward.

(Refer Slide Time: 16:46)



Loop gain under VMC

$$K_{loop}(s) = F_m \times G_{vd}(s) \times G_{PID}(s) = \left(F_m V_{in} K_i\right) \times \frac{1}{s} \times \frac{N_c(s)}{D_p(s)}$$

$F_m = \frac{1}{V_m}$

Under perfect stable pole zero cancellation

$$N_c(s) = D_p(s)$$

$$\textcircled{1}\ \frac{K_p}{K_i} = \frac{L}{R}$$

$$\textcircled{2}\ \frac{K_d}{K_i} = LC$$

Three unknowns, but two equations

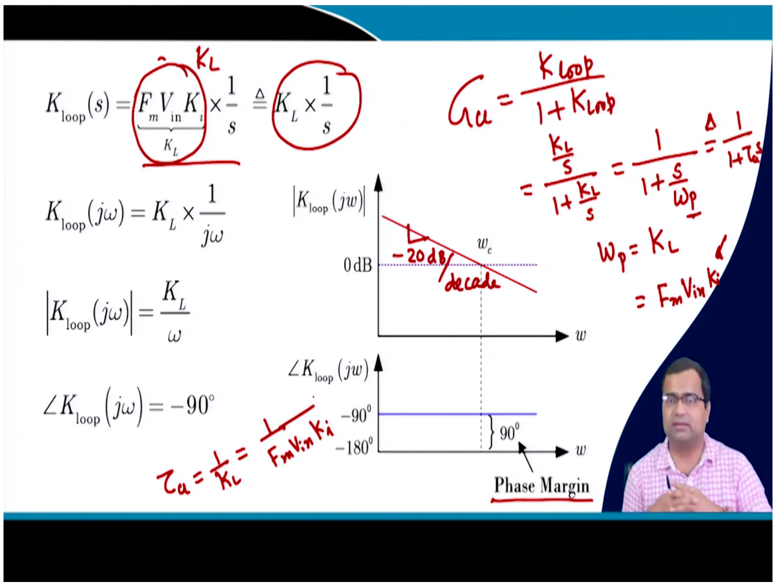
The next part is the loop gain under voltage mode control; it is the product of the modulator gain and modulator gain is nothing but 1 by V m, where V m is my the maximum voltage of the sawtooth waveform; that means, if I take the sawtooth waveform like this. So, I am it start from 0 and this is my V m ok.

So, this is a modulator gain. G vd is a control to duty ratio transfer of function, which is also called control to output, but here duty ratio is the control variable. So, it is a control to output transfer function where the duty ratio is the control input and this is our PID controller ok.

So, we can write this and numerator of the controller we discuss; that means, N c D p and under perfect stable pole zero cancellation. So, we can write down from the previous expression that K p by K i is equal to L by R and K d by R. So, if you go back so, previous. So, this is we are talking about N c, rather than I will say N con, it is a N c right. So, I can just rub this because ok. So, it is N c ok.

So, now, you see this L by R, simply k p by k r; that means, this is my L by R k p by k r and k d by k i is simply L C ok. So, that is what we have seen here? But, here there are two equations 1 and 2, but there are three unknown, 1, 2 and 3. So, we cannot solve it. Because, if there are two equation three unknown, we can get an infinite number of solution, depending upon if you choose K i some value you will get K p and K d. So, you can have an infinite set of values right.

(Refer Slide Time: 18:37)



$$K_{loop}(s) = \underbrace{F_m V_{in} K_i}_{K_L} \times \frac{1}{s} \triangleq K_L \times \frac{1}{s}$$

$$K_{loop}(j\omega) = K_L \times \frac{1}{j\omega}$$

$$\left|K_{loop}(j\omega)\right| = \frac{K_L}{\omega}$$

$$\angle K_{loop}(j\omega) = -90^{0}$$

$$G_{cl} = \frac{K_{loop}}{1+K_{loop}} = \frac{\frac{K_L}{s}}{1+\frac{K_L}{s}} = \frac{1}{1+\frac{s}{\omega_p}} \triangleq \frac{1}{1+\tau s}$$

$$\omega_p = K_L = F_m V_{in} K_i$$

$$\tau_{cl} = \frac{1}{K_L} = \frac{1}{F_m V_{in} K_i}$$

Then, the loop transfer function if you do perfect compensation, because we are cancelling the denominator of this transfer function by the numerator of this transfer function. So, the rest is nothing but this loop transfer function ok. Here, we are considering this whole thing to be K L. This is my K L.

And, if we do that K L the loop transfer function is simply is an integrator with a gain right. And, if you take the loop gain, magnitude of the loop gain is simply K L by omega and if you take the phase, it is minus 90 degrees. So, it is it looks like this; that means, this slope is simply minus 20 dB per decade in log scale like decade ok.

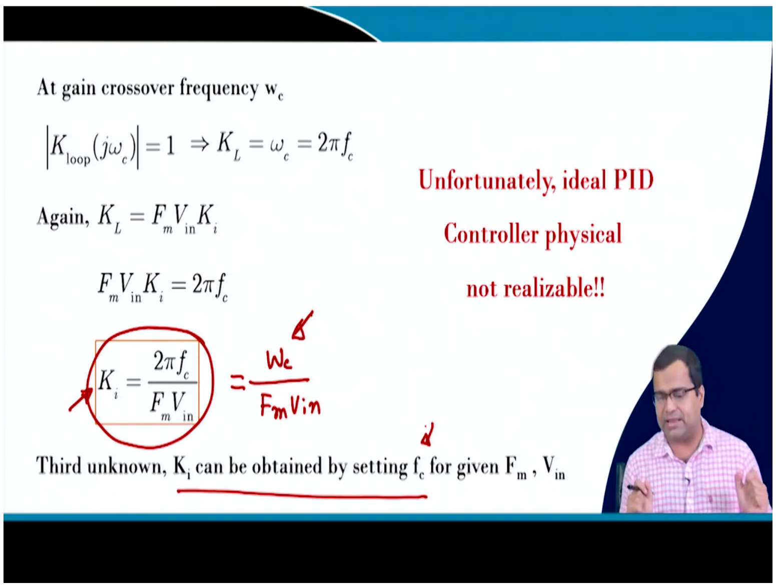
So, gain plot it is because it is just pure integrator and you have a phase of minus 90 degree. So, as a result, your phase margin is 90 degree. So, this phase margin seems to be quite large, because it will lead to over damp system, because your loop transfer function is a first order system.

So, naturally, if you want to obtain the closed loop transfer function, it is nothing but our loop transfer function by 1 plus loop transfer function ok. And what we will get. So, here it is K L by S, 1 by K L by S. So, we can write 1 by 1 plus S by omega p. So, where omega p I can write 1 by K L ok.

So, omega p is simply omega p is simply K L; that means you know if you write it will be s by K L, where K L is yes, omega p should be K L. And, what is K L here? So, we have written here right. So, it is nothing but F m V in K i; that means now we need to set what is my omega p, that is my design variable.

So, it is the only 1 degree of freedom; that means, by changing the integral gain and if we write in terms of time constant 1 by 1 plus s tau closed loop into s. Where tau closed loop is simply 1 by K L and which is nothing but 1 by F m v in into K i; that means, if you increase the integral gain a closed loop time constant decreases. So, it will respond faster. But now the question is that how far you can increase the K i.

(Refer Slide Time: 21:29)



At gain crossover frequency $w_c$

$$\left|K_{\text{loop}}(j\omega_c)\right| = 1 \Rightarrow K_L = \omega_c = 2\pi f_c$$

Again, $K_L = F_m V_{in} K_i$

$$F_m V_{in} K_i = 2\pi f_c$$

$$\boxed{K_i = \frac{2\pi f_c}{F_m V_{in}}} = \frac{W_c}{F_m V_{in}}$$

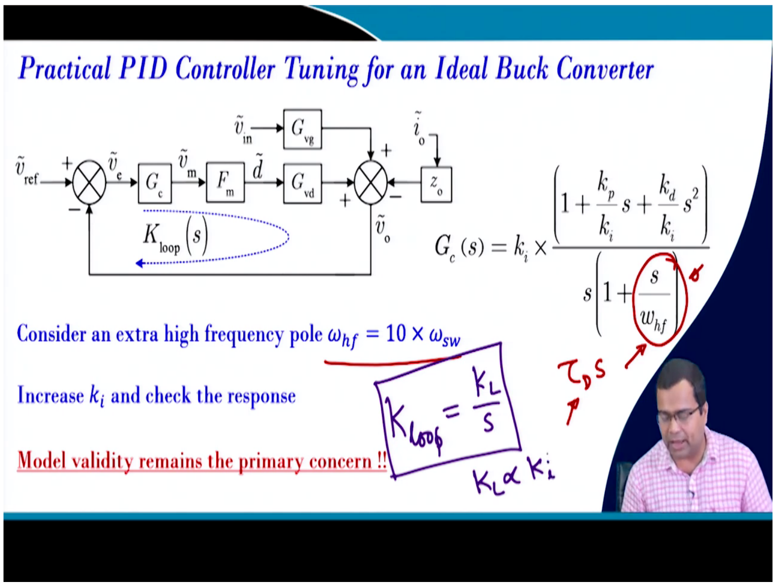
Third unknown, $K_i$ can be obtained by setting $f_c$ for given $F_m$, $V_{in}$

Unfortunately, ideal PID Controller physical not realizable!!

So, at crossover frequency, our gain loop gain magnitude will be 1 from there we can find out that, you know this K i can be obtained by this which is nothing but our in radian per second crossover frequency F m by V in ok. So, now the question is that, if you want to increase the crossover frequency, you have to simply increase the integral gain. And, it is the only degree of freedom, which can be used to change the cutoff frequency and which eventually decides the closed-loop bandwidth.

So, third unknown is here and K i to obtain by setting the crossover frequency and we want to for a given F m, V n, we want to see what crossover frequency we can achieve. But, unfortunately, it we have used an ideal PID controller, which cannot be physically realized.

(Refer Slide Time: 22:26)



## Practical PID Controller Tuning for an Ideal Buck Converter

$$G_c(s) = k_i \times \frac{\left(1 + \frac{k_p}{k_i}s + \frac{k_d}{k_i}s^2\right)}{s\left(1 + \frac{s}{w_{hf}}\right)}$$

Consider an extra high frequency pole $\omega_{hf} = 10 \times \omega_{sw}$

Increase $k_i$ and check the response

Model validity remains the primary concern !!

$$\boxed{K_{loop} = \frac{K_L}{s}}$$

$K_L \propto K_i$

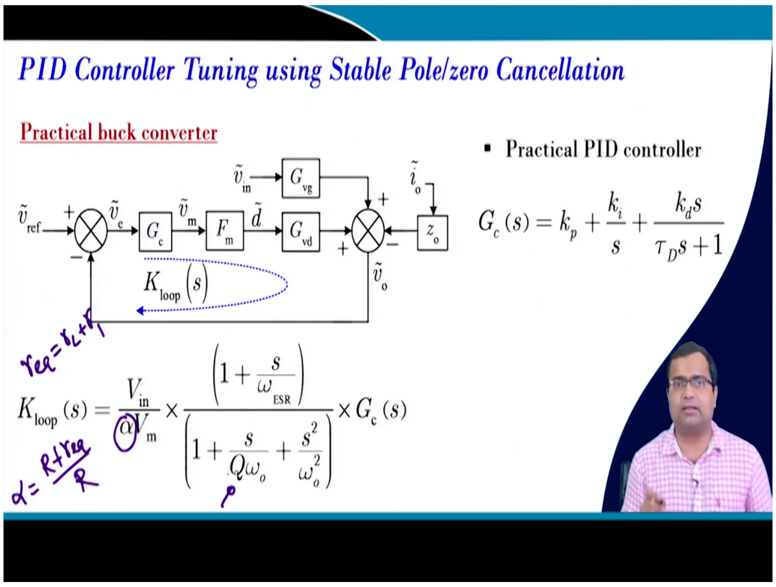
$\tau_D s$

So, in order to solve this problem, we need to consider a high frequency pole with the PID controller. So, which is equivalent to so, this particular term is equivalent to tau D into S where this was a derivative time constant right. And, I have written in terms of a pole. So, we are putting a high frequency pole and if we set this high frequency pole 10 time faster than switching frequency in radian per second.

Then, we can have a reduced effect of this pole into the actual closed loop performance, because this is just for sake of implementation of the PID controller. Otherwise, if you use an op-amp, the op-amp itself has a finite bandwidth and that this bandwidth will be set by the op-amp, ok.

So, you can increase the K i value and you can check the response and that we are going to do, because the same kind of structure will get even for a practical buck converter. And, we need to check the model validity. This part we hold on. Because, where what we got our loop transfer function that we obtain here. Our loop transfer function that we found that means, here we got the loop transfer function is K L by S. So, we want to see the closed loop performance for varying and where K L is a function of is proportional to K i ok.

(Refer Slide Time: 23:56)



## PID Controller Tuning using Stable Pole/zero Cancellation

Practical buck converter

- Practical PID controller

$$G_c(s) = k_p + \frac{k_i}{s} + \frac{k_d s}{\tau_D s + 1}$$

$$K_{loop}(s) = \frac{V_{in}}{\alpha V_m} \times \frac{\left(1 + \frac{s}{\omega_{ESR}}\right)}{\left(1 + \frac{s}{Q\omega_o} + \frac{s^2}{\omega_o^2}\right)} \times G_c(s)$$

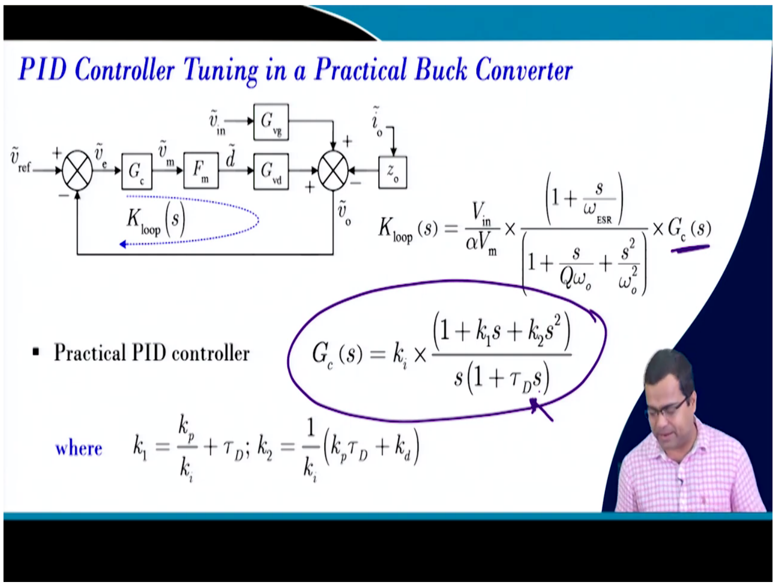
$r_{eq} = r_L + r_1$

$\alpha = \frac{R + r_{eq}}{R}$

The next we want to use PID controller for a practical buck converter, now it is a practical buck converter ok. Where we have considered practical buck converter, we have considered the actual parasitic; that means, you see this alpha is coming. What is alpha? The alpha we know it is nothing, but R plus r equivalent by R.

What is R equivalent? It is nothing but dcr plus rds 1, this we know. We talked about ESR, this Q factor and these things we have already discussed multiple time.

(Refer Slide Time: 24:34)



## PID Controller Tuning in a Practical Buck Converter

$$K_{loop}(s) = \frac{V_{in}}{\alpha V_m} \times \frac{\left(1 + \frac{s}{\omega_{ESR}}\right)}{\left(1 + \frac{s}{Q\omega_o} + \frac{s^2}{\omega_o^2}\right)} \times G_c(s)$$

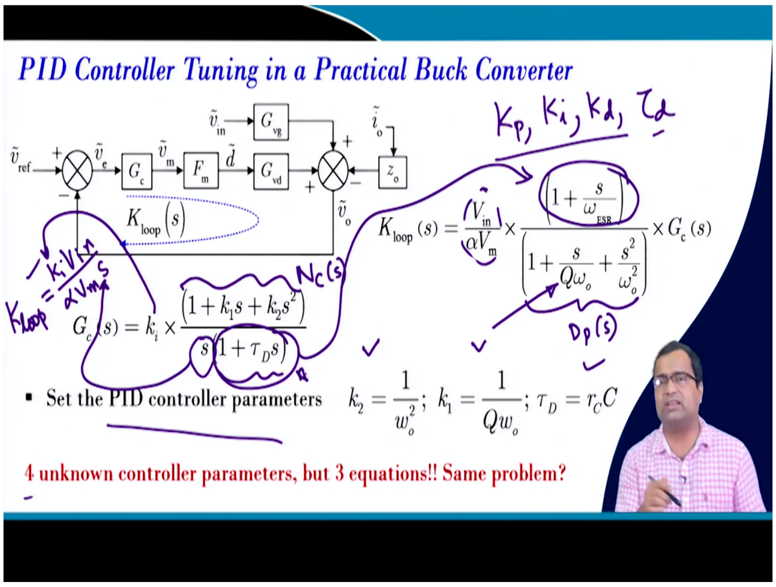
- Practical PID controller

$$G_c(s) = k_i \times \frac{\left(1 + k_1 s + k_2 s^2\right)}{s\left(1 + \tau_D s\right)}$$

where $k_1 = \frac{k_p}{k_i} + \tau_D$; $k_2 = \frac{1}{k_i}\left(k_p \tau_D + k_d\right)$

Now, we want to compensate. So, the loop transfer function of a practical buck converter, where we are trying to design this k c. And, we are taking a PID controller. This is a practical PID controller right, which has a derivative filter. So, practical PID controller and we know this k i k 1 k 2 in terms of k p k i k d.

(Refer Slide Time: 25:03)



## PID Controller Tuning in a Practical Buck Converter

$K_p, K_i, K_d, \tau_d$

$$K_{loop}(s) = \frac{V_{in}}{\alpha V_m} \times \frac{\left(1+\frac{s}{\omega_{ESR}}\right)}{\left(1+\frac{s}{Q\omega_o}+\frac{s^2}{\omega_o^2}\right)} \times G_c(s)$$

$$G_c(s) = k_i \times \frac{\left(1+k_1 s+k_2 s^2\right)}{s\left(1+\tau_D s\right)}$$

$N_c(s)$ $D_p(s)$

$K_{loop} = \frac{k_i V_{in}}{\alpha V_m s}$

- Set the PID controller parameters $k_2 = \frac{1}{w_o^2}$; $k_1 = \frac{1}{Q w_o}$; $\tau_D = r_C C$

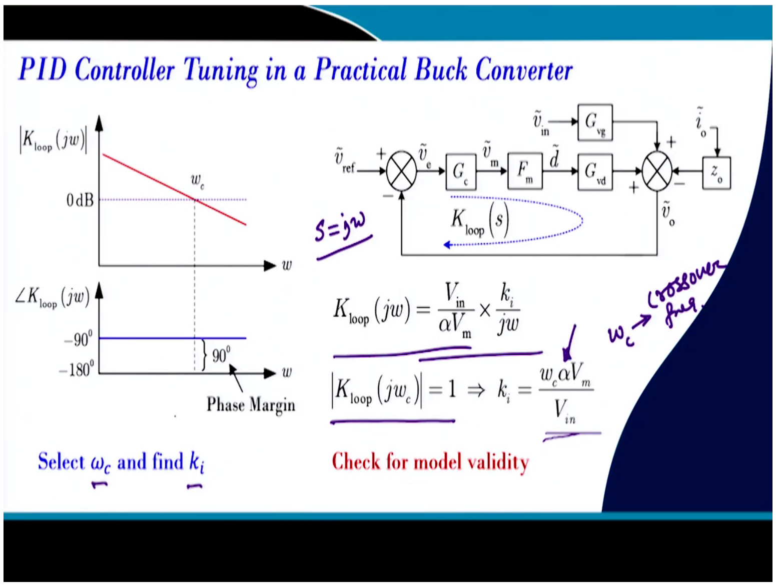
4 unknown controller parameters, but 3 equations!! Same problem?

Now, for this transfer function, if we set the PID controller; that means, here again we want to the numerator coefficient we want to cancel with denominator polynomial. And, since they are stable, we can do stable pole 0 cancellation. But, again it is a non-robust choice because the Q is nothing but it is a function of the resistance and which may not be a robust solution and that will discuss in the next lecture.

So, now if you set it then, we have 4 unknown? Because how many unknowns? Is 1 is K p K i; that means, what are the unknown? We have K p K i K d and tau d. Because, in the first case we have only three unknown for ideal, where we kept these two be very low, which we do not need to find out, because it can be kept much lower. Because we have said that the frequency corresponding to this will be much higher than the switching frequency.

But, here we have four unknown and this is one equation, this is second equation, and third equation. So, we have 3 equations, so, same problem.

(Refer Slide Time: 26:17)



## PID Controller Tuning in a Practical Buck Converter

$$K_{loop}(jw) = \frac{V_{in}}{\alpha V_m} \times \frac{k_i}{jw}$$

$$|K_{loop}(jw_c)| = 1 \Rightarrow k_i = \frac{w_c \alpha V_m}{V_{in}}$$

Select $\omega_c$ and find $k_i$

Check for model validity

But, if we compensate again after this pole zero cancellation. Now, we are setting this ESR 0 and this term; that means, if we take this term we are going to cancel with this term. That means we are canceling ESR 0 which can be reasonable, which is reasonably known by using the derivative filter pole.

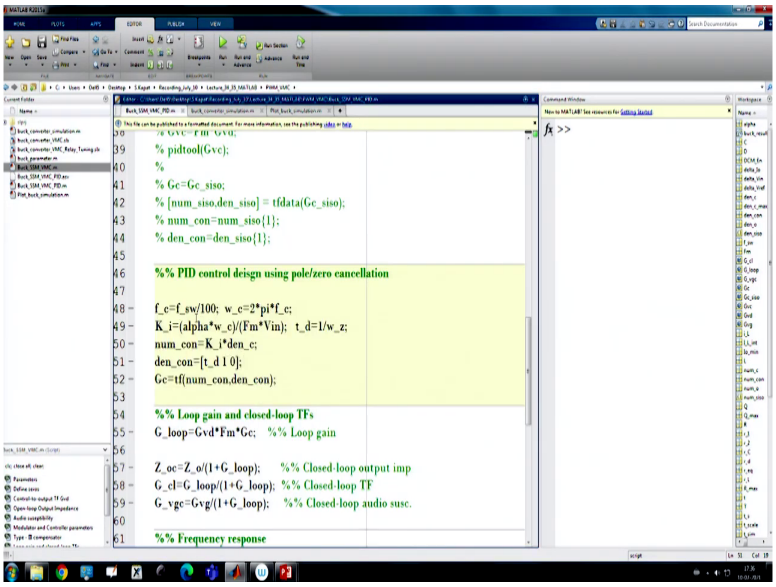
So, there is a pole zero cancellation. The controller 2 0s are used to cancel the planned pole ok. So, the rest part is that that means, if you write the loop transfer function. What is our loop transfer function now? After this compensation, our loop transfer function is simply you know K i this, K i then V in alpha V m by S. Because, here we have an 1 by s term it is coming here. The K i term is coming here, and this term is coming here ok.

Now, if you write the frequency response, we replace s equal to j omega, then we are writing k i V in alpha V m. So, this is the expression. So, this is same as earlier except this term is come this comes into picture. And, if your r equivalent is very small, this alpha will be 1. So, it will be same as the earlier.

But, here we have not considered a derivative filter. We have not canceled the ESR. And, since the ESR is very slow that it has ESR 0 will be at high frequency. So, this is also a high frequency pole. So, they can be used to cancel each other.

Then, the resultant loop transfer function again becomes a pure integrator same as earlier. And, we can select omega c and k i from this crossover frequency, because this is at crossover frequency. So, this is omega c is our crossover frequency right. So, crossover frequency, we can find it out and now we have to check the model validity.

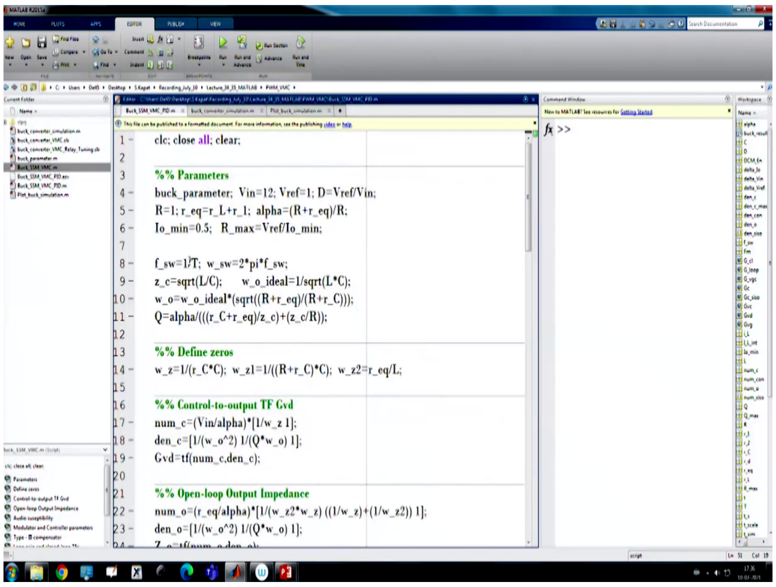
(Refer Slide Time: 28:45)



```
% pidtool(Gvc);
%
% Gc=Gc_siso;
% [num_siso,den_siso] = tfdata(Gc_siso);
% num_con=num_siso{1};
% den_con=den_siso{1};

%% PID control deisgn using pole/zero cancellation

f_c=f_sw/100;  w_c=2*pi*f_c;
K_i=(alpha*w_c)/(Fm*Vin);   t_d=1/w_z;
num_con=K_i*den_c;
den_con=[t_d 1 0];
Gc=tf(num_con,den_con);

%% Loop gain and closed-loop TFs
G_loop=Gvd*Fm*Gc;    %% Loop gain

Z_oc=Z_o/(1+G_loop);      %% Closed-loop output imp
G_cl=G_loop/(1+G_loop);  %% Closed-loop TF
G_vgc=Gvg/(1+G_loop);    %% Closed-loop audio susc.

%% Frequency response
```

So, let us go to the MATLAB, what we are doing here?

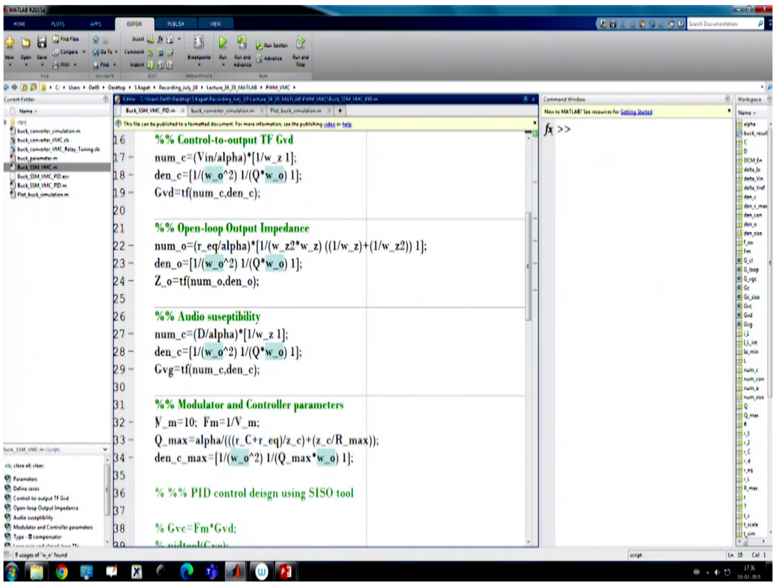
(Refer Slide Time: 28:51)



```
clc; close all; clear;

%% Parameters
buck_parameter;  Vin=12; Vref=1; D=Vref/Vin;
R=1; r_eq=r_L+r_l;  alpha=(R+r_eq)/R;
Io_min=0.5;   R_max=Vref/Io_min;

f_sw=1/T;  w_sw=2*pi*f_sw;
z_c=sqrt(L/C);      w_o_ideal=1/sqrt(L*C);
w_o=w_o_ideal*(sqrt((R+r_eq)/(R+r_C)));
Q=alpha/(((r_C+r_eq)/z_c)+(z_c/R));

%% Define zeros
w_z=1/(r_C*C);  w_z1=1/((R+r_C)*C);  w_z2=r_eq/L;

%% Control-to-output TF Gvd
num_c=(Vin/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvd=tf(num_c,den_c);

%% Open-loop Output Impedance
num_o=(r_eq/alpha)*[1/(w_z2*w_z) ((1/w_z)+(1/w_z2)) 1];
den_o=[1/(w_o^2) 1/(Q*w_o) 1];
```

Here, we are taking again the small signal model and we have discussed in the previous class. That parameter file you know we have all this Q calculation omega 0, all these calculations you know we have discussed, while you know validating the small signal model.

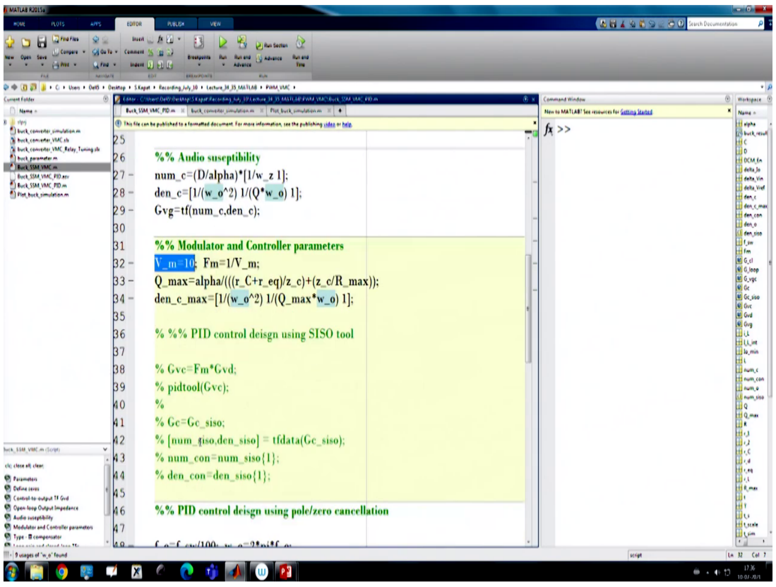
(Refer Slide Time: 29:00)



```
%% Control-to-output TF Gvd
num_c=(Vin/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvd=tf(num_c,den_c);

%% Open-loop Output Impedance
num_o=(r_eq/alpha)*[1/(w_z2*w_z) ((1/w_z)+(1/w_z2)) 1];
den_o=[1/(w_o^2) 1/(Q*w_o) 1];
Z_o=tf(num_o,den_o);

%% Audio suseptibility
num_c=(D/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvg=tf(num_c,den_c);

%% Modulator and Controller parameters
V_m=10;  Fm=1/V_m;
Q_max=alpha/(((r_C+r_eq)/z_c)+(z_c/R_max));
den_c_max=[1/(w_o^2) 1/(Q_max*w_o) 1];

% %% PID control deisgn using SISO tool

% Gvc=Fm*Gvd;
```

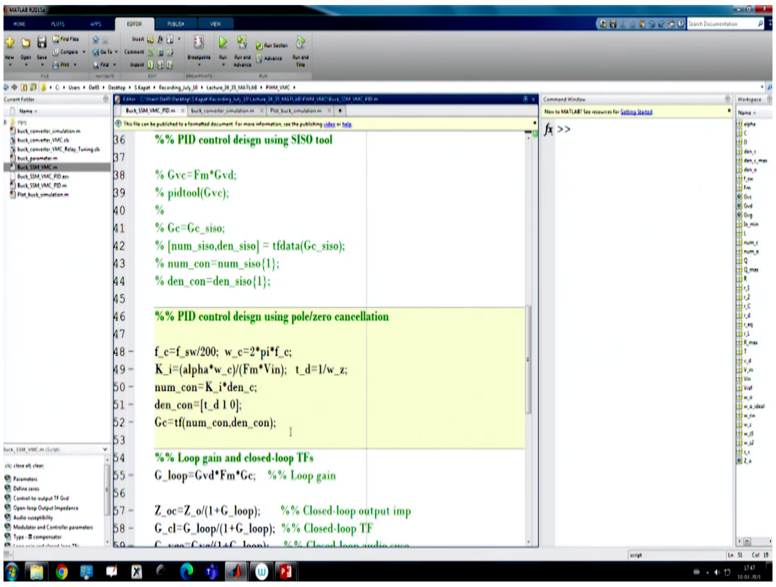
So, audio susceptibility open loop output impedance control to output transfer function everything, the modulator voltage is set as 10 volt.

(Refer Slide Time: 29:15)



```

%% Audio suseptibility
num_c=(D/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvg=tf(num_c,den_c);

%% Modulator and Controller parameters
V_m=10;  Fm=1/V_m;
Q_max=alpha/(((r_C+r_eq)/z_c)+(z_c/R_max));
den_c_max=[1/(w_o^2) 1/(Q_max*w_o) 1];

% %% PID control deisgn using SISO tool

% Gvc=Fm*Gvd;
% pidtool(Gvc);
%
% Gc=Gc_siso;
% [num_siso,den_siso] = tfdata(Gc_siso);
% num_con=num_siso{1};
% den_con=den_siso{1};

%% PID control deisgn using pole/zero cancellation

```

(Refer Slide Time: 29:16)

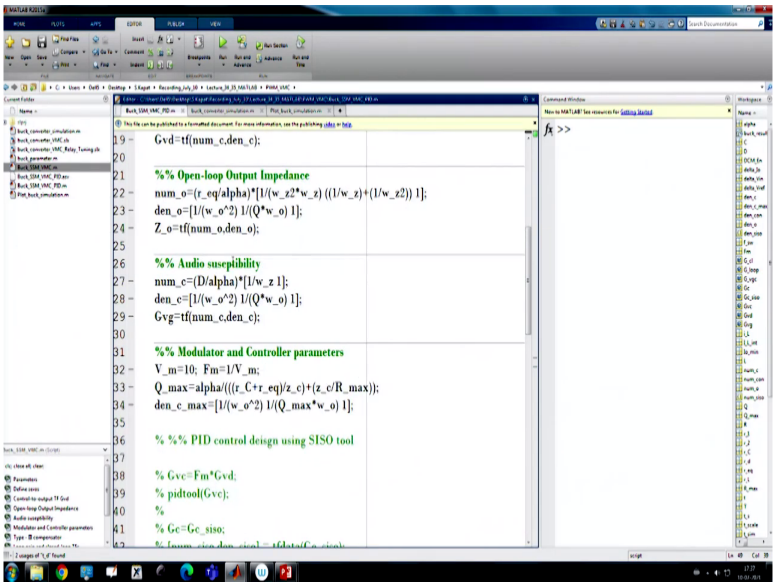


```
%% PID control deisgn using SISO tool

% Gvc=Fm*Gvd;
% pidtool(Gvc);
%
% Gc=Gc_siso;
% [num_siso,den_siso] = tfdata(Gc_siso);
% num_con=num_siso{1};
% den_con=den_siso{1};

%% PID control deisgn using pole/zero cancellation

f_c=f_sw/200;  w_c=2*pi*f_c;
K_i=(alpha*w_c)/(Fm*Vin);   t_d=1/w_z;
num_con=K_i*den_c;
den_con=[t_d 1 0];
Gc=tf(num_con,den_con);

%% Loop gain and closed-loop TFs
G_loop=Gvd*Fm*Gc;    %% Loop gain

Z_oc=Z_o/(1+G_loop);      %% Closed-loop output imp
G_cl=G_loop/(1+G_loop);  %% Closed-loop TF
```

And, then this is an analytical method of PID controller design, where cut off frequency is set to 100 times. It is much lower, but I will show you why I said it ok. Then, omega c can be computed, because if you go back and you will see that omega c here we are writing, it is nothing but 2 pi FC right.

So, it is coming from here alpha V m by in. So, it is coming here K i equal to alpha omega c divided by F m V in and t d the derivative filter time constant is 1 by ESR 0 which is omega z here. You see, the omega z is used ESR 0 ok.

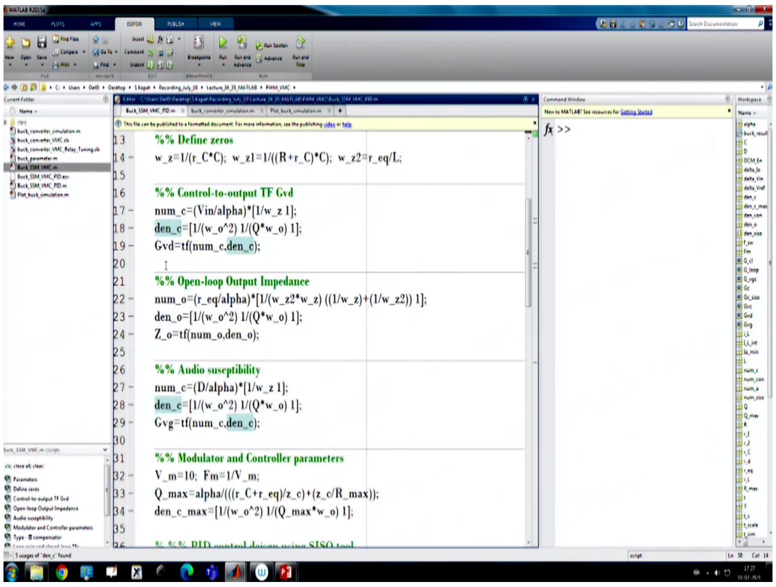
(Refer Slide Time: 29:55)



```
Gvd=tf(num_c,den_c);

%% Open-loop Output Impedance
num_o=(r_eq/alpha)*[1/(w_z2*w_z) ((1/w_z)+(1/w_z2)) 1];
den_o=[1/(w_o^2) 1/(Q*w_o) 1];
Z_o=tf(num_o,den_o);

%% Audio susceptibility
num_c=(D/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvg=tf(num_c,den_c);

%% Modulator and Controller parameters
V_m=10;  Fm=1/V_m;
Q_max=alpha/(((r_C+r_eq)/z_c)+(z_c/R_max));
den_c_max=[1/(w_o^2) 1/(Q_max*w_o) 1];

% %% PID control deisgn using SISO tool

% Gvc=Fm*Gvd;
% pidtool(Gvc);
%
% Gc=Gc_siso;
```

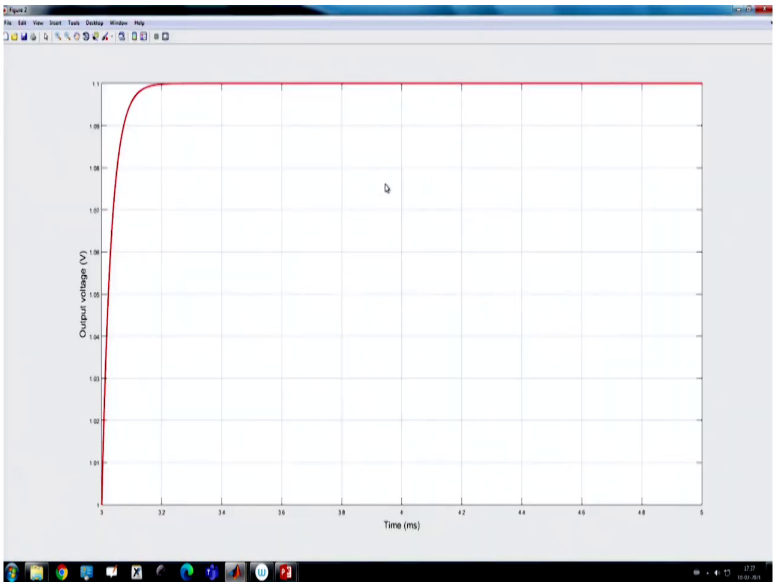
So, it is a ESR 0. Then, we have you know numerator and denominator. Denominator of this controller is taken as the numerator sorry numerator of the controller is K i into the d p; that means, the denominator of the plane which is nothing but the denominator of this control to output transfer function ok.

(Refer Slide Time: 30:16)



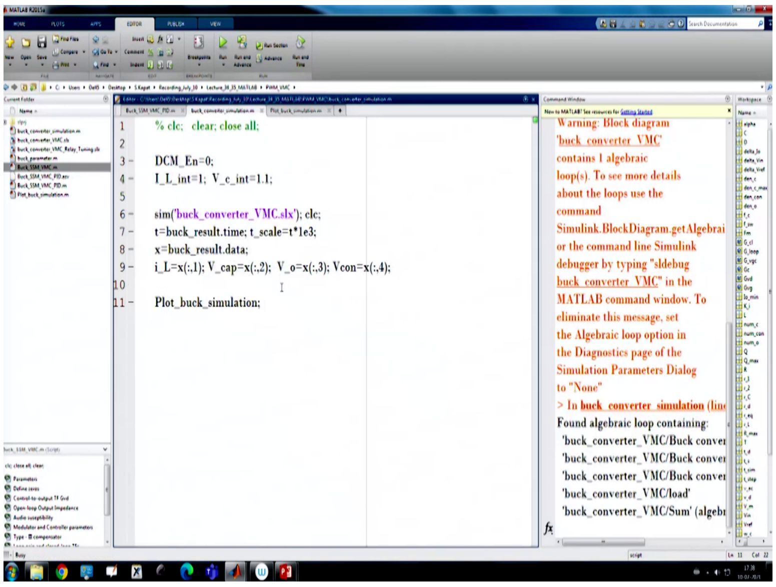
```
%% Define zeros
w_z=1/(r_C*C);  w_z1=1/((R+r_C)*C);  w_z2=r_eq/L;

%% Control-to-output TF Gvd
num_c=(Vin/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvd=tf(num_c,den_c);

%% Open-loop Output Impedance
num_o=(r_eq/alpha)*[1/(w_z2*w_z) ((1/w_z)+(1/w_z2)) 1];
den_o=[1/(w_o^2) 1/(Q*w_o) 1];
Z_o=tf(num_o,den_o);

%% Audio suseptibility
num_c=(D/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvg=tf(num_c,den_c);

%% Modulator and Controller parameters
V_m=10;  Fm=1/V_m;
Q_max=alpha/(((r_C+r_eq)/z_c)+(z_c/R_max));
den_c_max=[1/(w_o^2) 1/(Q_max*w_o) 1];

% %% PID control deisgn using SISO tool
```

(Refer Slide Time: 30:23)



So, now, with this let us simulate and check. So, this is the response of the system over damp system, and we have applied a point 1 volt reference voltage transient. So, initially the output voltage was 1 volt, now is changed from 1 volt to 1.1 volt ok.

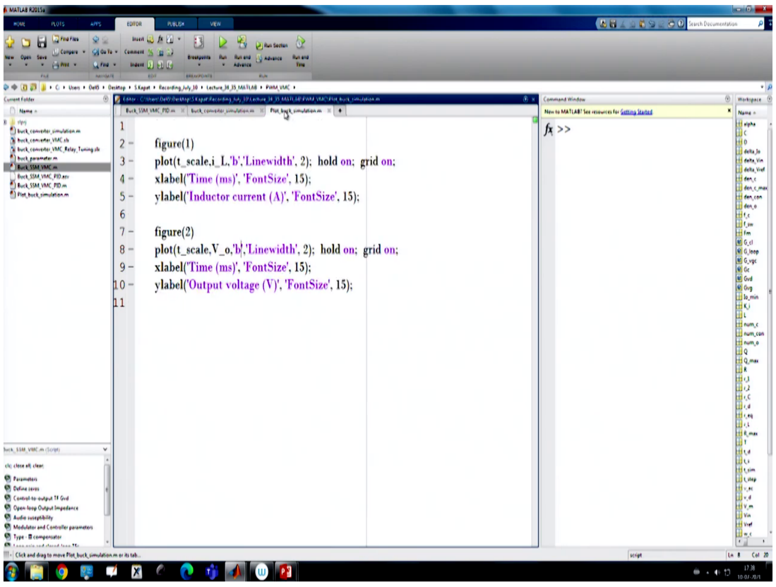
(Refer Slide Time: 30:42)



```
% clc;  clear; close all;

DCM_En=0;
I_L_int=1;  V_c_int=1.1;

sim('buck_converter_VMC.slx'); clc;
t=buck_result.time; t_scale=t*1e3;
x=buck_result.data;
i_L=x(:,1); V_cap=x(:,2);  V_o=x(:,3); Vcon=x(:,4);

Plot_buck_simulation;
```

```
Warning: Block diagram
'buck_converter_VMC'
contains 1 algebraic
loop(s). To see more details
about the loops use the
command
Simulink.BlockDiagram.getAlgebrai
or the command line Simulink
debugger by typing "sldebug
buck_converter_VMC" in the
MATLAB command window. To
eliminate this message, set
the Algebraic loop option in
the Diagnostics page of the
Simulation Parameters Dialog
to "None"
> In buck_converter_simulation (lin
Found algebraic loop containing:
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/load'
 'buck_converter_VMC/Sum' (algebr
```

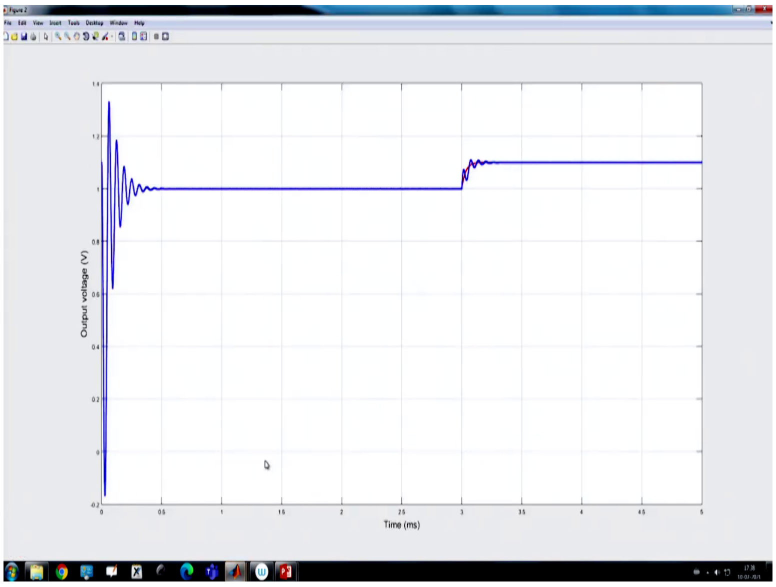
(Refer Slide Time: 30:43)



```
figure(1)
plot(t_scale,i_L,'b','Linewidth', 2);  hold on;  grid on;
xlabel('Time (ms)', 'FontSize', 15);
ylabel('Inductor current (A)', 'FontSize', 15);

figure(2)
plot(t_scale,V_o,'b','Linewidth', 2);  hold on;  grid on;
xlabel('Time (ms)', 'FontSize', 15);
ylabel('Output voltage (V)', 'FontSize', 15);
```

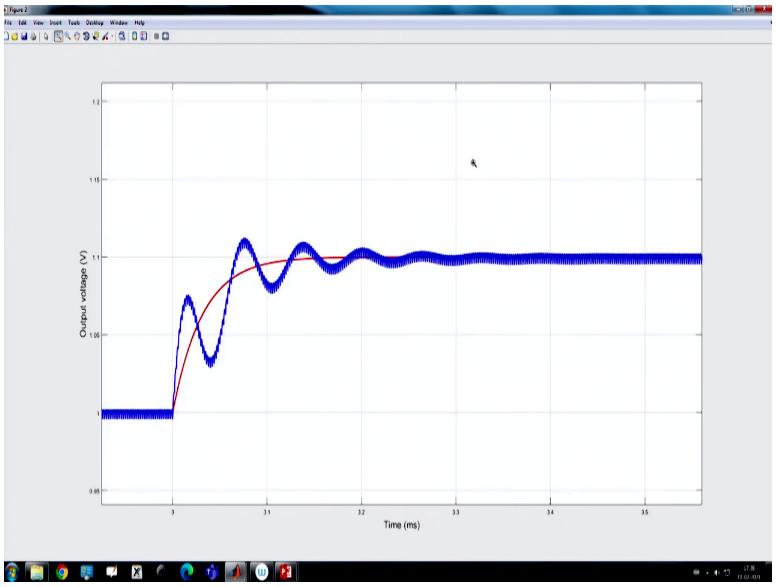
Now, we want to see the response matching response with our small signal model. So, let us run it.

(Refer Slide Time: 30:49)



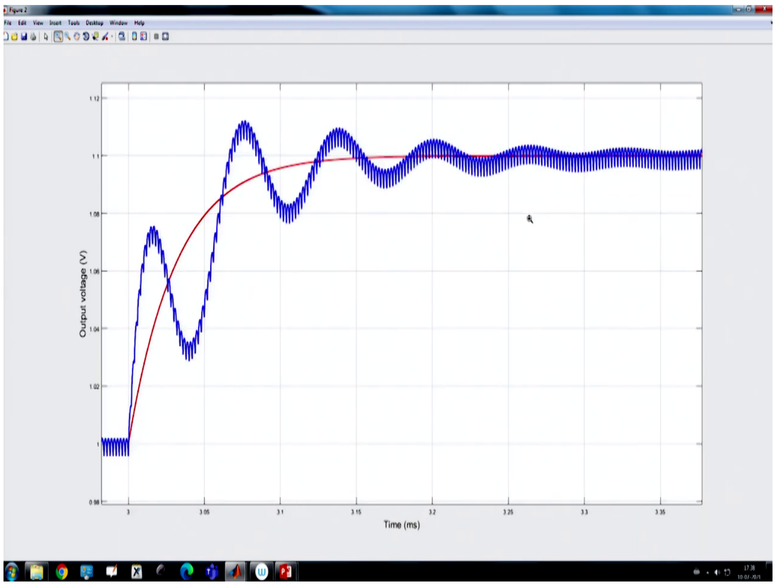
And, we want to verify whether they ok.

(Refer Slide Time: 30:54)



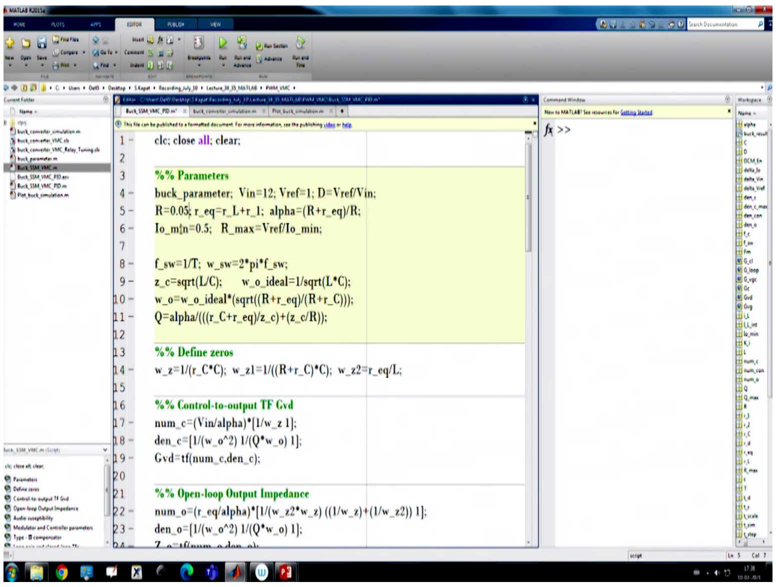
You will find that, the response is not at all matching ok.

(Refer Slide Time: 30:56)



So, there is a significant deviation from your first-order model and from your actual, you know, response from the small signal model. So, in order to do that, let us do because we have taken the load resistance to be very low.

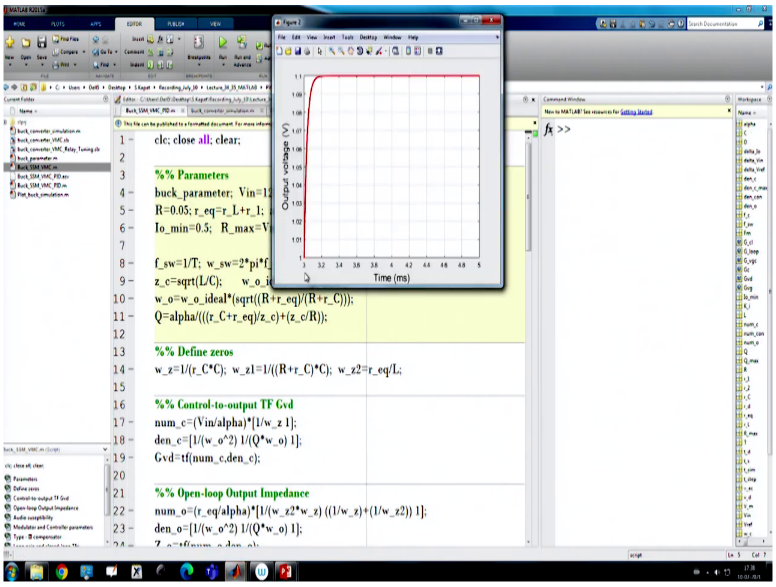
(Refer Slide Time: 31:14)



```
clc; close all; clear;

%% Parameters
buck_parameter; Vin=12; Vref=1; D=Vref/Vin;
R=0.05; r_eq=r_L+r_1; alpha=(R+r_eq)/R;
Io_min=0.5;  R_max=Vref/Io_min;

f_sw=1/T;  w_sw=2*pi*f_sw;
z_c=sqrt(L/C);      w_o_ideal=1/sqrt(L*C);
w_o=w_o_ideal*(sqrt((R+r_eq)/(R+r_C)));
Q=alpha/(((r_C+r_eq)/z_c)+(z_c/R));

%% Define zeros
w_z=1/(r_C*C);  w_z1=1/((R+r_C)*C);  w_z2=r_eq/L;

%% Control-to-output TF Gvd
num_c=(Vin/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvd=tf(num_c,den_c);

%% Open-loop Output Impedance
num_o=(r_eq/alpha)*[1/(w_z2*w_z) ((1/w_z)+(1/w_z2)) 1];
den_o=[1/(w_o^2) 1/(Q*w_o) 1];
```

So, if we take 0.05 at 20 ampere load.

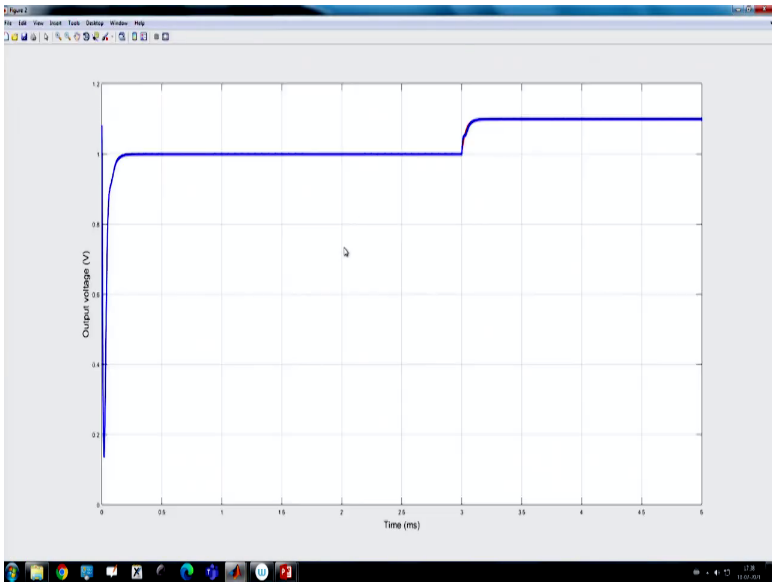
(Refer Slide Time: 31:19)



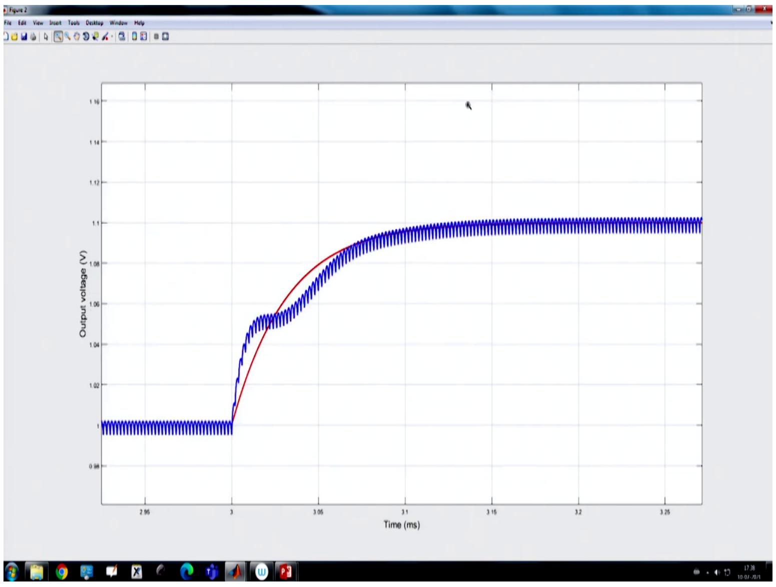
```
clc; close all; clear;

%% Parameters
buck_parameter;  Vin=12
R=0.05; r_eq=r_L+r_1;
Io_min=0.5;   R_max=V

f_sw=1/T;  w_sw=2*pi*f_
z_c=sqrt(L/C);      w_o_i
w_o=w_o_ideal*(sqrt((R+r_eq)/(R+r_C)));
Q=alpha/(((r_C+r_eq)/z_c)+(z_c/R));

%% Define zeros
w_z=1/(r_C*C);  w_z1=1/((R+r_C)*C);  w_z2=r_eq/L;

%% Control-to-output TF Gvd
num_c=(Vin/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvd=tf(num_c,den_c);

%% Open-loop Output Impedance
num_o=(r_eq/alpha)*[1/(w_z2*w_z) ((1/w_z)+(1/w_z2)) 1];
den_o=[1/(w_o^2) 1/(Q*w_o) 1];
```

That means, load ok. So, this is the response and now again we are plotting, the same method we are following ok.

(Refer Slide Time: 31:26)

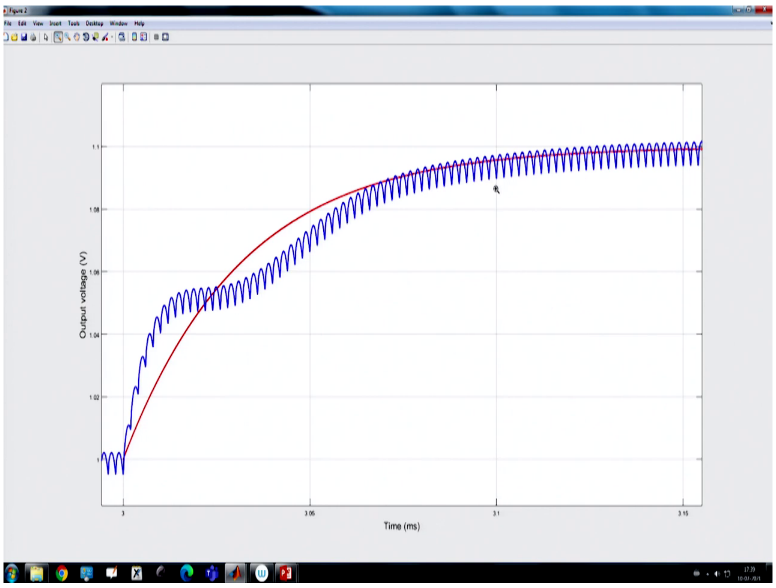


(Refer Slide Time: 31:32)



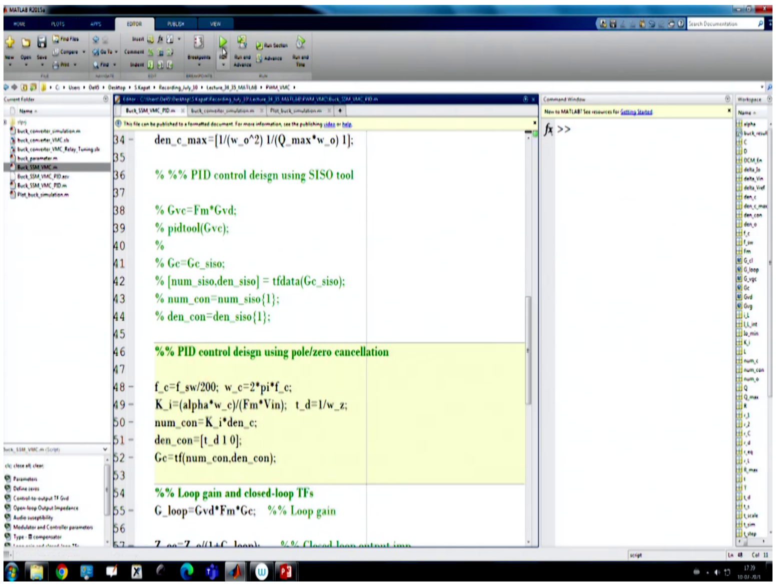
So, since we have increase you know.

(Refer Slide Time: 31:34)



So, now they are somewhat closely matching, but they are still not matching perfectly. Now, if we reduce further the bandwidth; that means, here we set the cutoff frequency.

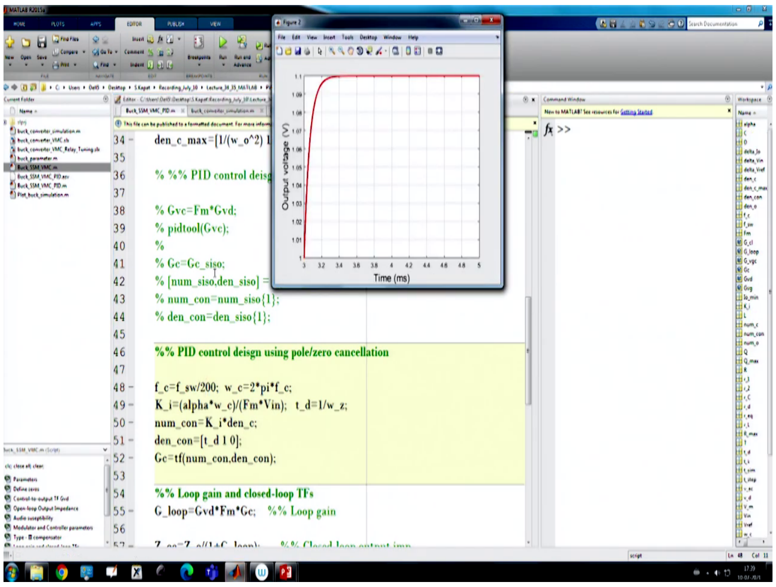
(Refer Slide Time: 31:44)



```
den_c_max=[1/(w_o^2) 1/(Q_max*w_o) 1];

% %% PID control deisgn using SISO tool

% Gvc=Fm*Gvd;
% pidtool(Gvc);
%
% Gc=Gc_siso;
% [num_siso,den_siso] = tfdata(Gc_siso);
% num_con=num_siso{1};
% den_con=den_siso{1};

%% PID control deisgn using pole/zero cancellation

f_c=f_sw/200;  w_c=2*pi*f_c;
K_i=(alpha*w_c)/(Fm*Vin);  t_d=1/w_z;
num_con=K_i*den_c;
den_con=[t_d 1 0];
Gc=tf(num_con,den_con);

%% Loop gain and closed-loop TFs
G_loop=Gvd*Fm*Gc;   %% Loop gain
```

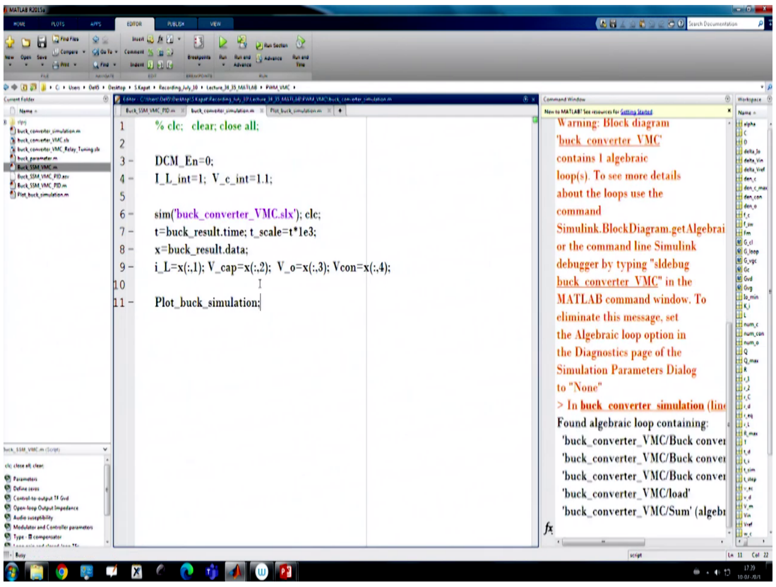
So, if you make 200 it is much slower.

(Refer Slide Time: 31:48)



```
f_c=f_sw/200;  w_c=2*pi*f_c;
K_i=(alpha*w_c)/(Fm*Vin);  t_d=1/w_z;
num_con=K_i*den_c;
den_con=[t_d 1 0];
Gc=tf(num_con,den_con);

%% Loop gain and closed-loop TFs
G_loop=Gvd*Fm*Gc;   %% Loop gain
```

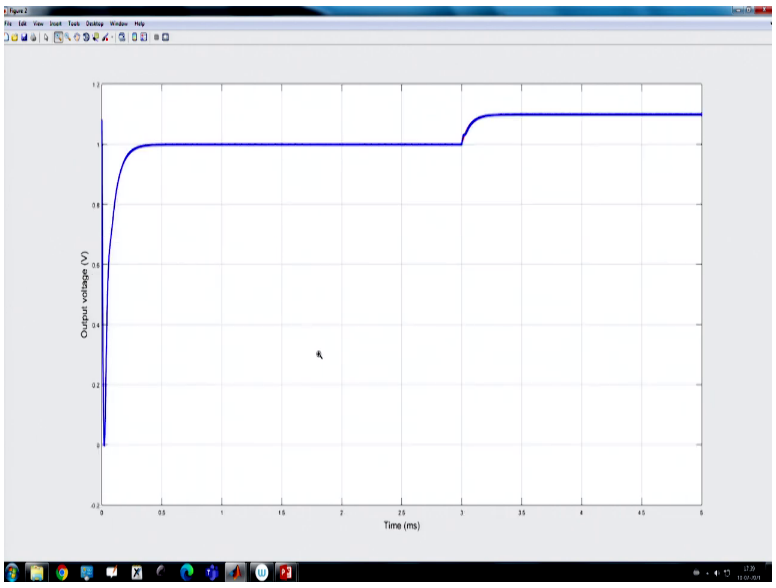
(Refer Slide Time: 31:51)



```
% clc;   clear; close all;

DCM_En=0;
I_L_int=1;  V_c_int=1.1;

sim('buck_converter_VMC.slx'); clc;
t=buck_result.time; t_scale=t*1e3;
x=buck_result.data;
i_L=x(:,1); V_cap=x(:,2);  V_o=x(:,3); Vcon=x(:,4);

Plot_buck_simulation;
```

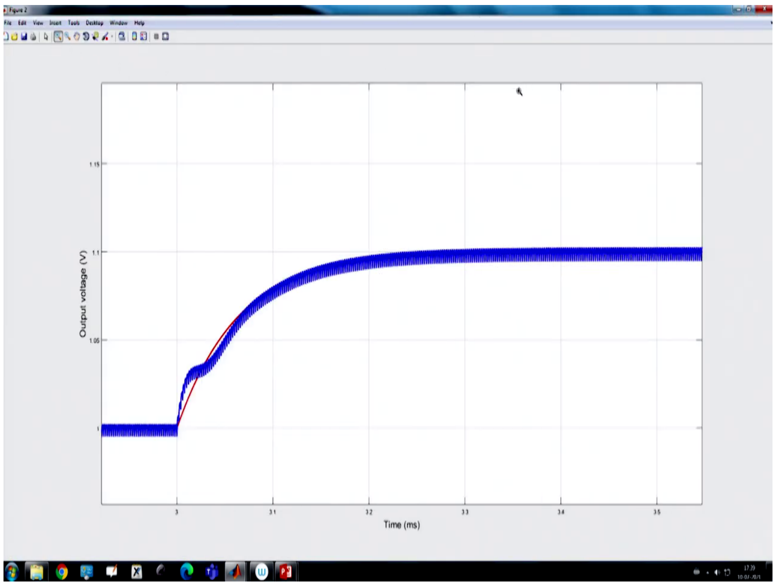
```
Warning: Block diagram
'buck_converter_VMC'
contains 1 algebraic
loop(s). To see more details
about the loops use the
command
Simulink.BlockDiagram.getAlgebrai
or the command line Simulink
debugger by typing "sldebug
buck_converter_VMC" in the
MATLAB command window. To
eliminate this message, set
the Algebraic loop option in
the Diagnostics page of the
Simulation Parameters Dialog
to "None"
> In buck_converter_simulation (line
Found algebraic loop containing:
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/load'
 'buck_converter_VMC/Sum' (algebr
```

(Refer Slide Time: 31:53)

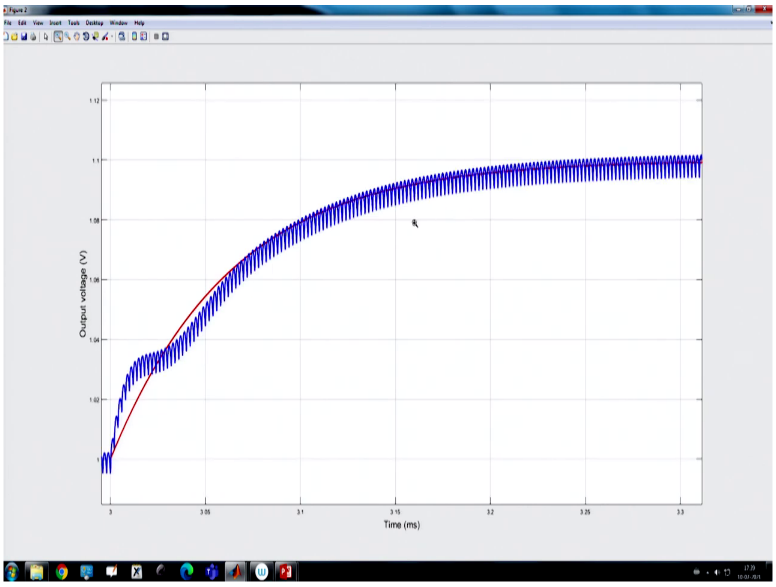


And, if you run it and if we try to match the response, then you will see they are matching closely.

(Refer Slide Time: 31:58)

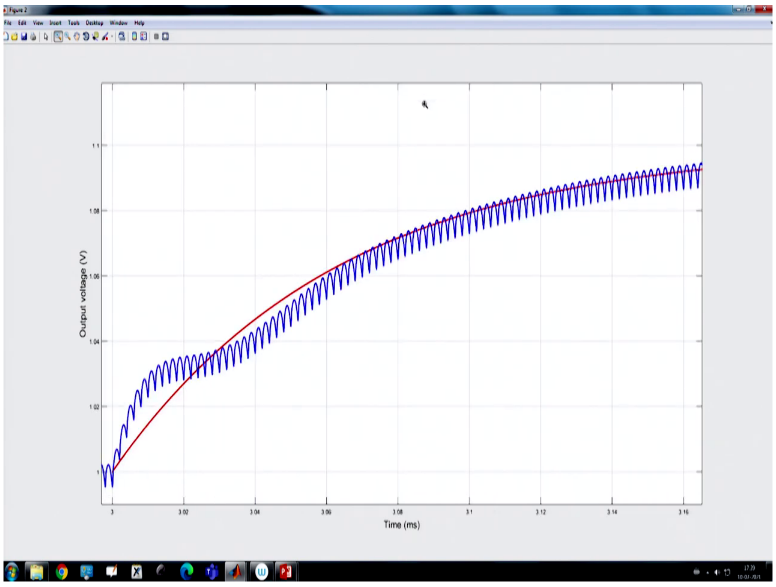


(Refer Slide Time: 32:00)



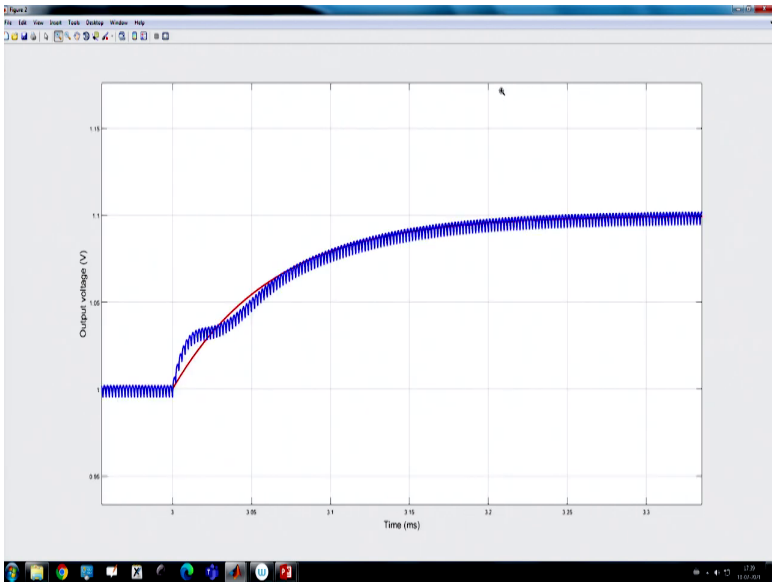
That means, this first-order model we cannot go for a higher bandwidth. Because this model is no longer valid, it cannot even capture the behavior properly. It is the time constant, because there is a deviation of the actual switch simulation ok.

(Refer Slide Time: 32:14)



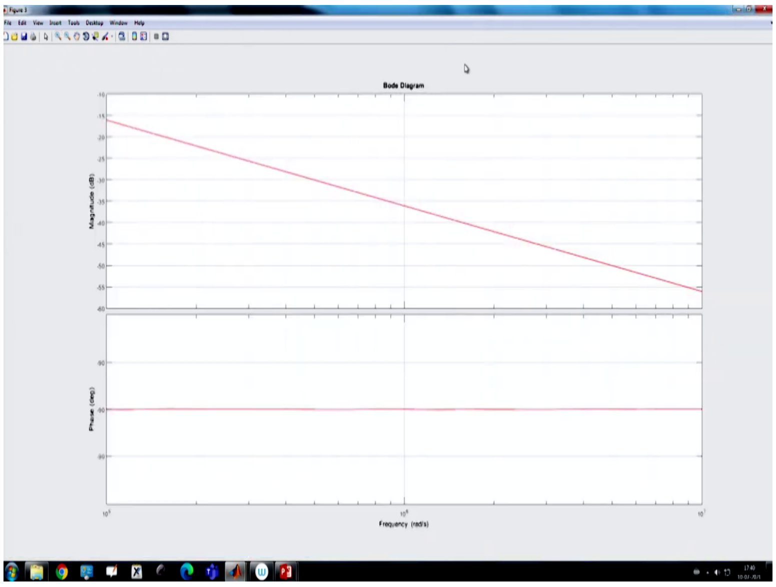
So, the model is not perfectly valid.

(Refer Slide Time: 32:18)



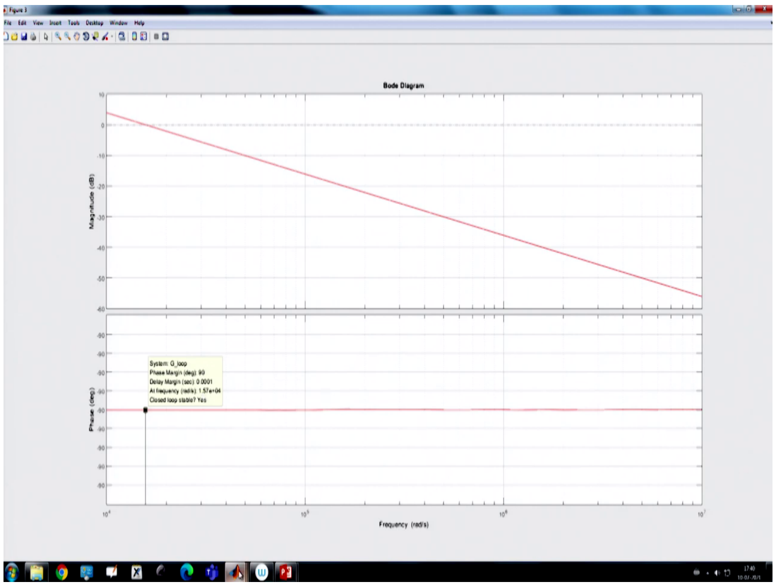
But, still it reasonably capture, but if you try to because we are even setting the cutoff frequency to be 1 by 200 times of the switching, which is very low ok.

(Refer Slide Time: 32:32)



So, it is too low.

(Refer Slide Time: 32:37)

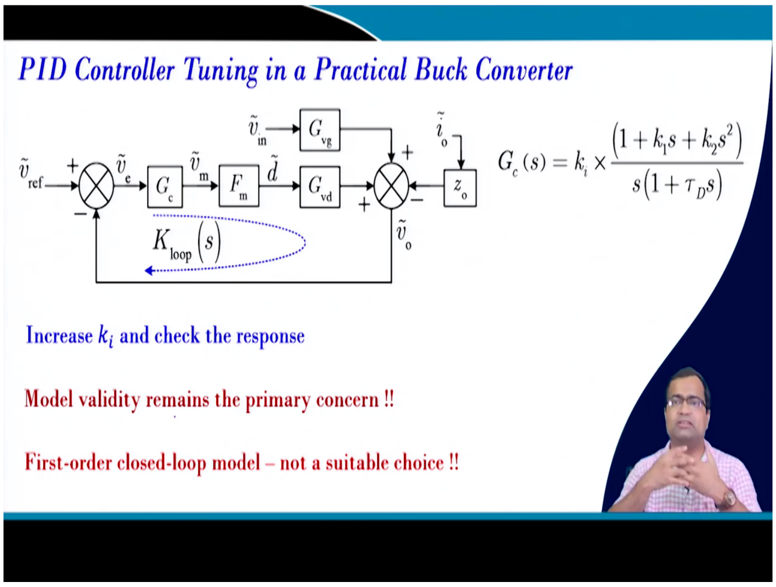


System: G_loop
Phase Margin (deg): 90
Delay Margin (sec): 0.0001
At frequency (rad/s): 1.57e+04
Closed loop stable? Yes

And, you see you the Bode plot. So, if you see the all the stability margin. You will see that the stability margin if you check, the phase margin is 90 degrees, crossover frequency is 1.57 maybe 15.7 kilohertz, which is too low right. So, this is something not acceptable. Even with this, we are unable to match the model ok; that means, our model is not valid.

So, whether you take an ideal buck or a practical buck, but a PID controller if you want to compensate and to obtain a first-order model, it is something is not sufficient.

(Refer Slide Time: 33:10)



## PID Controller Tuning in a Practical Buck Converter

$$G_c(s) = k_i \times \frac{(1 + k_1 s + k_2 s^2)}{s(1 + \tau_D s)}$$

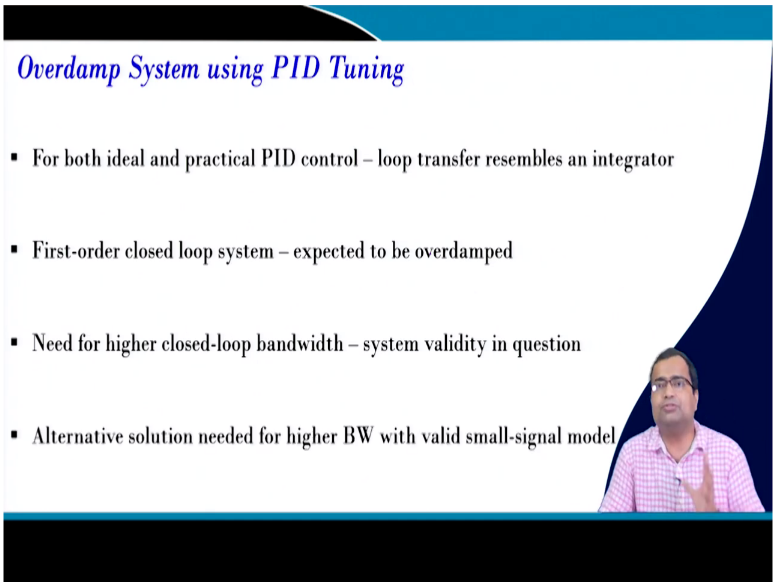
Increase $k_i$ and check the response

Model validity remains the primary concern !!

First-order closed-loop model – not a suitable choice !!

So, if you increase your bandwidth, the model validity is a concern. So, first order closed-loop model is not a suitable solid solution.

(Refer Slide Time: 33:19)

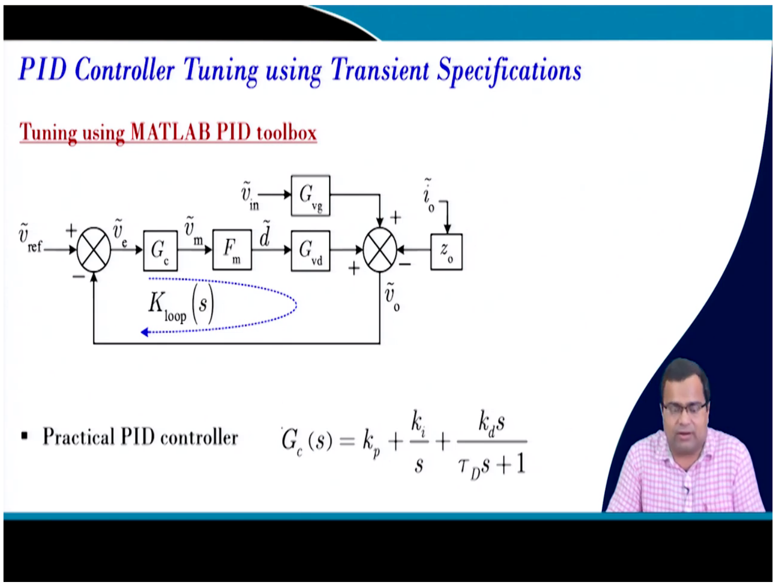


## Overdamp System using PID Tuning

- For both ideal and practical PID control – loop transfer resembles an integrator
- First-order closed loop system – expected to be overdamped
- Need for higher closed-loop bandwidth – system validity in question
- Alternative solution needed for higher BW with valid small-signal model

Now, so, both ideal and practical loop transfer, loop transfer function. Actually, resemble an integrator first order closed loop system expected to be overdamped, and that we got from the small signal model. But, if you want to increase the bandwidth a little bit the system validity remains in question. So, alternative solution needs to be found.

(Refer Slide Time: 33:41)



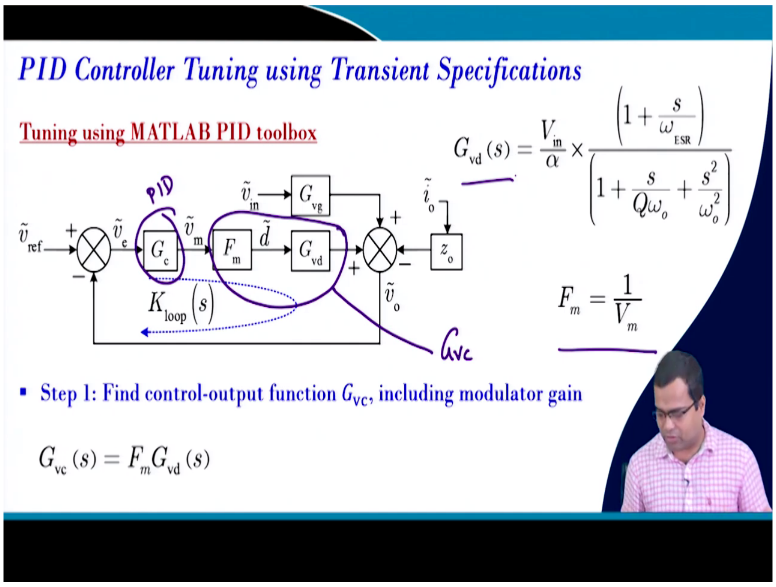
## PID Controller Tuning using Transient Specifications

Tuning using MATLAB PID toolbox

- Practical PID controller $G_c(s) = k_p + \frac{k_i}{s} + \frac{k_d s}{\tau_D s + 1}$

Now, we are going for the PID controller tuning using transient specification. So, here we are using MATLAB PID toolbox, ok. So, again, the same converter practical converter PID controller.

(Refer Slide Time: 33:57)



## PID Controller Tuning using Transient Specifications

Tuning using MATLAB PID toolbox

$$G_{vd}(s) = \frac{V_{in}}{\alpha} \times \frac{\left(1 + \frac{s}{\omega_{ESR}}\right)}{\left(1 + \frac{s}{Q\omega_o} + \frac{s^2}{\omega_o^2}\right)}$$

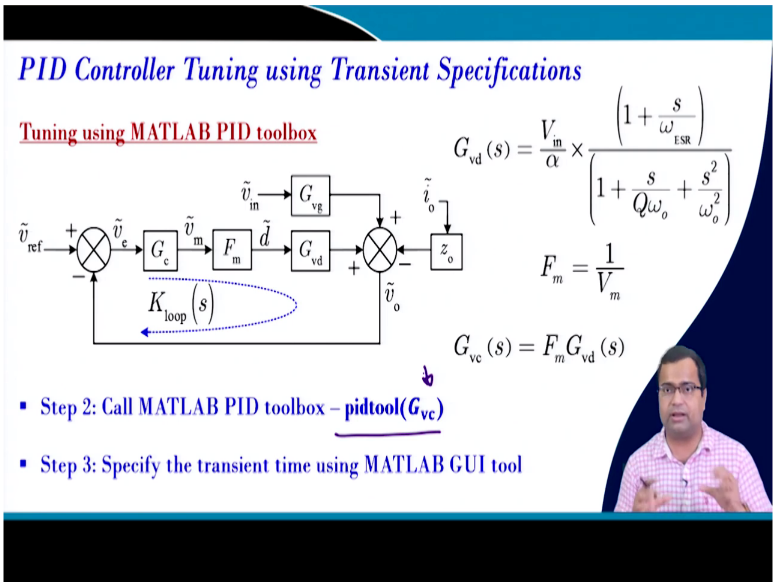
$$F_m = \frac{1}{V_m}$$

- Step 1: Find control-output function $G_{vc}$, including modulator gain

$$G_{vc}(s) = F_m G_{vd}(s)$$

But, now we are getting so, first we have what we are doing we want to obtain G vc and; that means, the control to because we are trying to find out this PID controller. But, we need this whole transfer function. And, this we are calling G vc, which is nothing but modulator gained G vd ok. So, the modulator gain expression is this and G vd is this they are practical converter.

(Refer Slide Time: 34:24)



## PID Controller Tuning using Transient Specifications

Tuning using MATLAB PID toolbox

$\tilde{v}_{ref}$
$\tilde{v}_e$
$G_c$
$\tilde{v}_m$
$F_m$
$\tilde{d}$
$G_{vd}$
$\tilde{v}_{in}$
$G_{vg}$
$\tilde{i}_o$
$z_o$
$\tilde{v}_o$
$K_{loop}(s)$

$$G_{vd}(s) = \frac{V_{in}}{\alpha} \times \frac{\left(1+\frac{s}{\omega_{ESR}}\right)}{\left(1+\frac{s}{Q\omega_o}+\frac{s^2}{\omega_o^2}\right)}$$

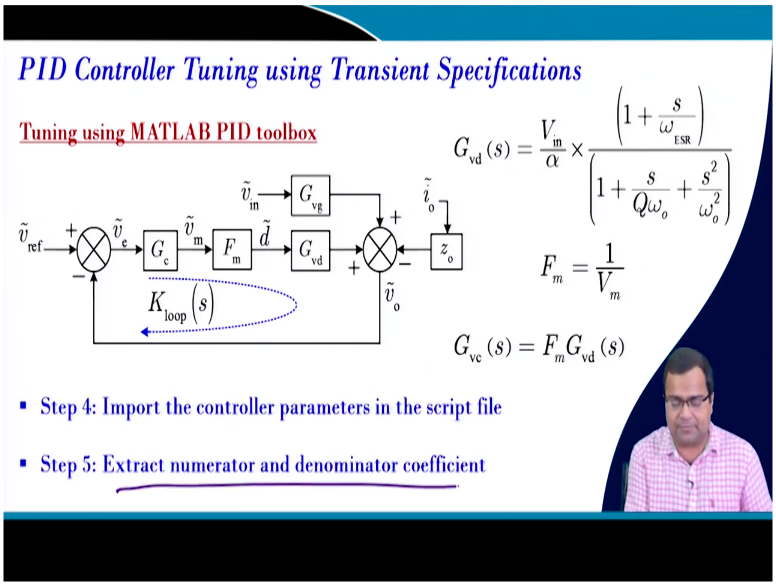
$$F_m = \frac{1}{V_m}$$

$$G_{vc}(s) = F_m G_{vd}(s)$$

- Step 2: Call MATLAB PID toolbox – **pidtool($G_{vc}$)**
- Step 3: Specify the transient time using MATLAB GUI tool

Then, what we have to do? We have to call the MATLAB PID toolbox. So, I am just first telling the summary before you go back to the actual demonstration using MATLAB. So, you have to call this PID toolbox and this transfer function is defined already assuming. Then, we need to specify the transient time using the MATLAB graphical interface tool.

(Refer Slide Time: 34:46)



## PID Controller Tuning using Transient Specifications

Tuning using MATLAB PID toolbox

$$G_{vd}(s) = \frac{V_{in}}{\alpha} \times \frac{\left(1+\frac{s}{\omega_{ESR}}\right)}{\left(1+\frac{s}{Q\omega_o}+\frac{s^2}{\omega_o^2}\right)}$$

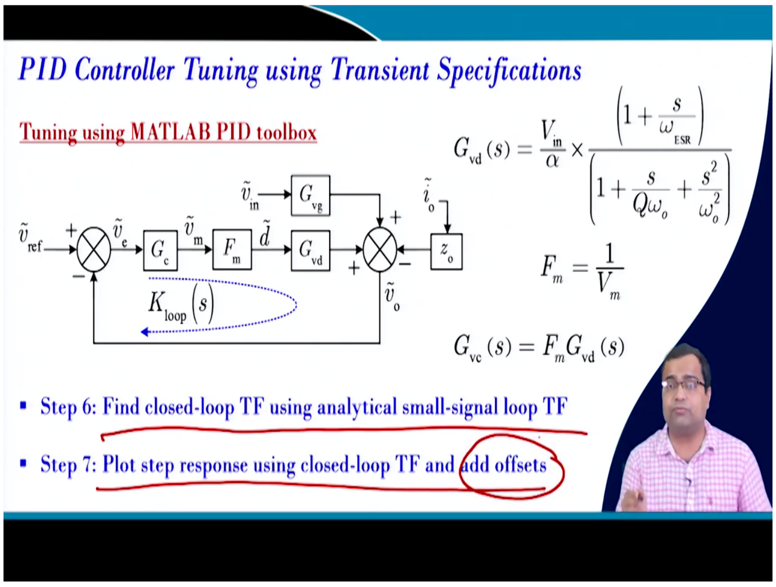
$$F_m = \frac{1}{V_m}$$

$$G_{vc}(s) = F_m G_{vd}(s)$$

- Step 4: Import the controller parameters in the script file
- Step 5: Extract numerator and denominator coefficient

And, then we have to import the controller parameter in the skip file, then extract the numerator and denominator coefficient, and then that mean numerator and denominator coefficient ok.

(Refer Slide Time: 35:00)



## PID Controller Tuning using Transient Specifications

Tuning using MATLAB PID toolbox

$$G_{vd}(s) = \frac{V_{in}}{\alpha} \times \frac{\left(1+\frac{s}{\omega_{ESR}}\right)}{\left(1+\frac{s}{Q\omega_o}+\frac{s^2}{\omega_o^2}\right)}$$

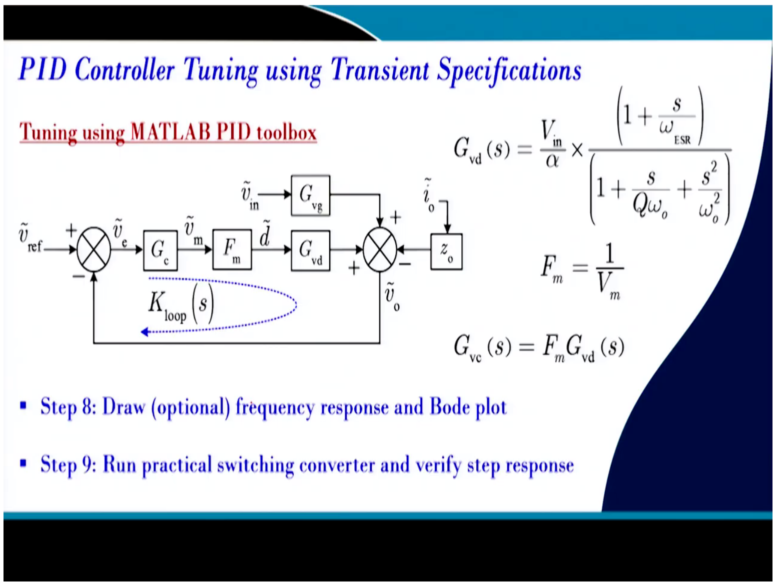
$$F_m = \frac{1}{V_m}$$

$$G_{vc}(s) = F_m G_{vd}(s)$$

- Step 6: Find closed-loop TF using analytical small-signal loop TF
- Step 7: Plot step response using closed-loop TF and add offsets

Then, find the closed loop transfer function using analytical model loop transfer function; plot the response using our transfer function. And, this is a ac response that we know and we have to add offset, because we want to verify with the actual switch simulation.

(Refer Slide Time: 35:17)



## PID Controller Tuning using Transient Specifications

Tuning using MATLAB PID toolbox

$$G_{vd}(s) = \frac{V_{in}}{\alpha} \times \frac{\left(1+\frac{s}{\omega_{ESR}}\right)}{\left(1+\frac{s}{Q\omega_o}+\frac{s^2}{\omega_o^2}\right)}$$

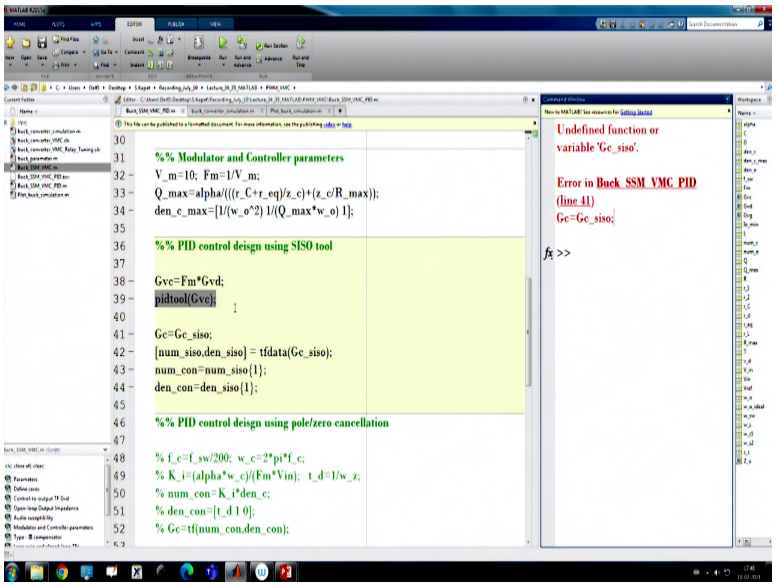
$$F_m = \frac{1}{V_m}$$

$$G_{vc}(s) = F_m G_{vd}(s)$$

- Step 8: Draw (optional) frequency response and Bode plot
- Step 9: Run practical switching converter and verify step response

Then, we draw we want to draw the frequency response and the Bode plot and then run the practical switching converter and verify the response ok. So, we want to now go for the actual MATLAB simulation, ok.

(Refer Slide Time: 35:29)



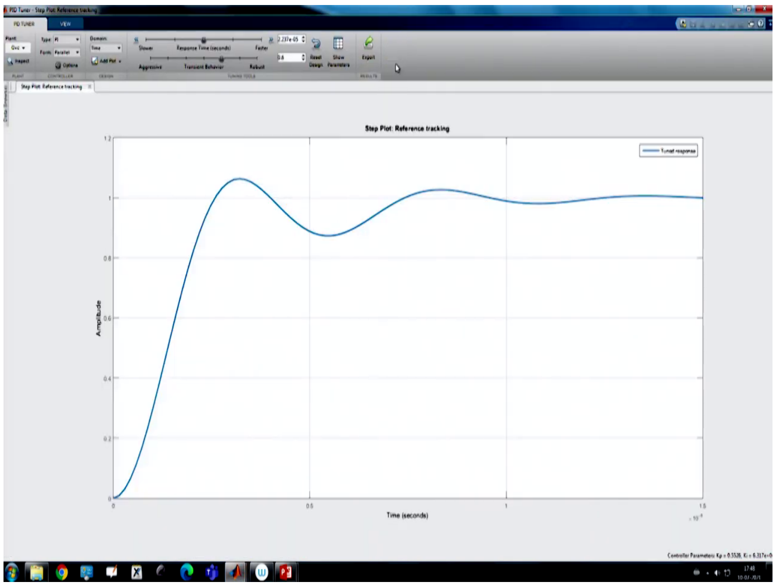
```
30
31    %% Modulator and Controller parameters
32 -  V_m=10;  Fm=1/V_m;
33 -  Q_max=alpha/(((r_C+r_eq)/z_c)+(z_c/R_max));
34 -  den_c_max=[1/(w_o^2) 1/(Q_max*w_o) 1];
35
36    %% PID control deisgn using SISO tool
37
38 -  Gvc=Fm*Gvd;
39 -  pidtool(Gvc);
40
41 -  Gc=Gc_siso;
42 -  [num_siso,den_siso] = tfdata(Gc_siso);
43 -  num_con=num_siso{1};
44 -  den_con=den_siso{1};
45
46    %% PID control deisgn using pole/zero cancellation
47
48    % f_c=f_sw/200;  w_c=2*pi*f_c;
49    % K_i=(alpha*w_c)/(Fm*Vin);   t_d=1/w_z;
50    % num_con=K_i*den_c;
51    % den_con=[t_d 1 0];
52    % Gc=tf(num_con,den_con);
```

```
Undefined function or
variable 'Gc_siso'.

Error in Buck_SSM_VMC_PID
(line 41)
Gc=Gc_siso;

fx >>
```

So, earlier we talked about PID controller design using pole zero cancellation, but now we want to go for the PID controller design using transient specification using MATLAB toolbox. So, this is the 1 that we are going to start discuss ok, this is ok. So, this is the 1 we are going to un common.

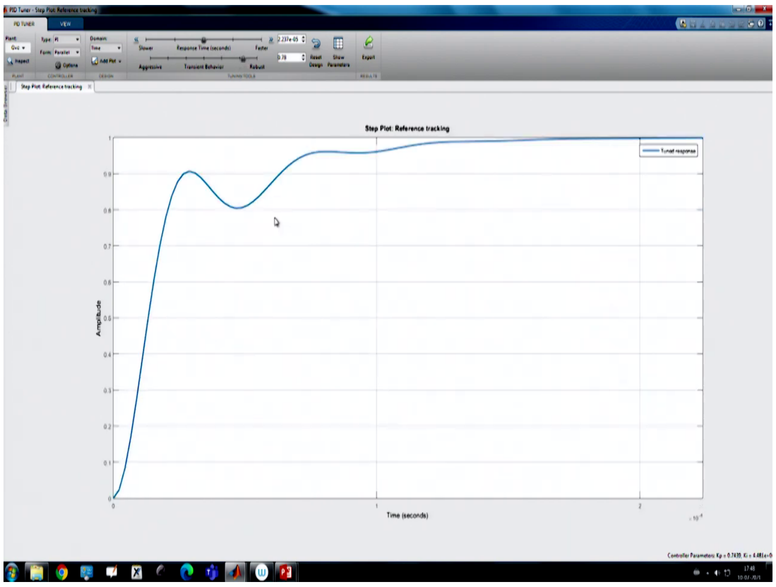
You can see, we need a G vc that we have discussed and PID tool ok and then let us run it. So, if we run, it you see it will show error because it enter into the G ui tool and it require this controller transfer function and do that we have to import from the GUI tool unless we import it this part cannot be executed. So, it is successfully executed up to this part ok. And, we will run the rest of the part once we import the parameter.

(Refer Slide Time: 36:27)



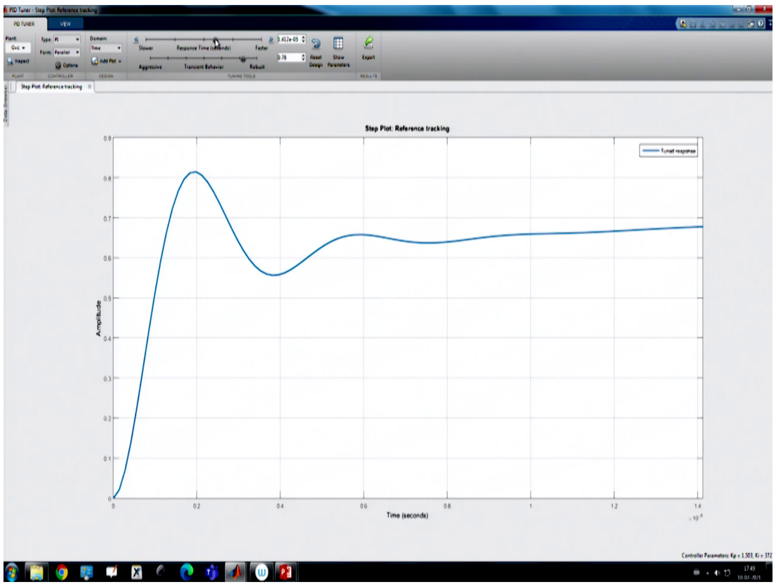
Now, let us go to the G ui tool. So, here we are going to set the transient specifications, ok.

(Refer Slide Time: 36:32)



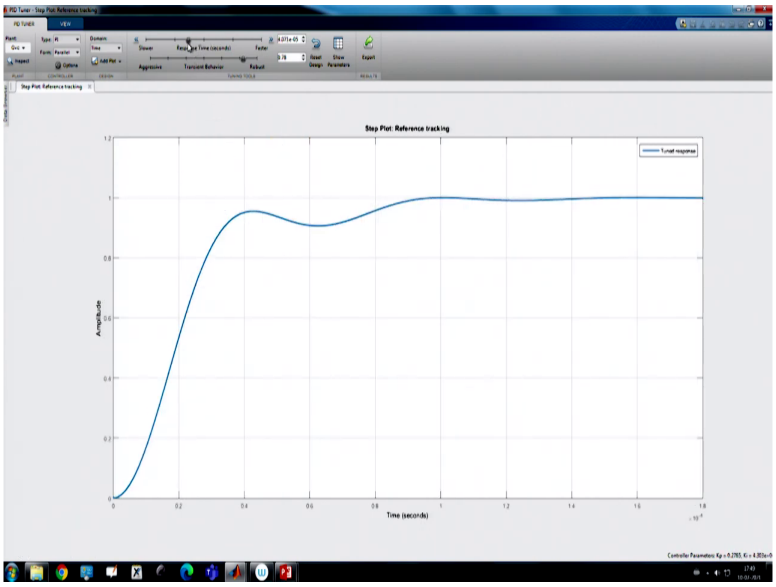
So, let us try to set the transient specification. Here you can see the time it is in the second. But, there is a 10 to the power minus 4; that means, if we talk about 1 into 10 to the power minus 4 second; that means it is 100 micro second ok, 100 micro second, because you want if you multiply by 100 10 to the power minus 6, 100 micro second. So, we are talking about 100 micro second.

(Refer Slide Time: 37:04)



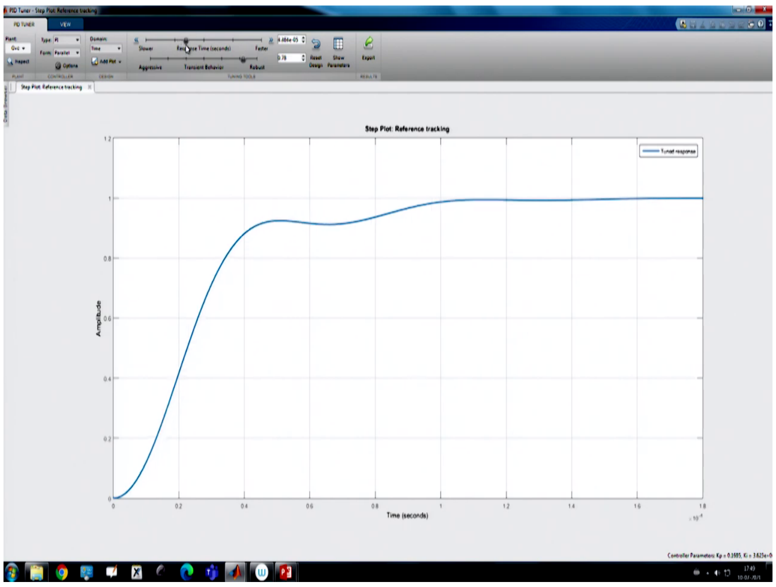
And, we need to achieve the response; that means, we want to make a bit faster ok.

(Refer Slide Time: 37:08)



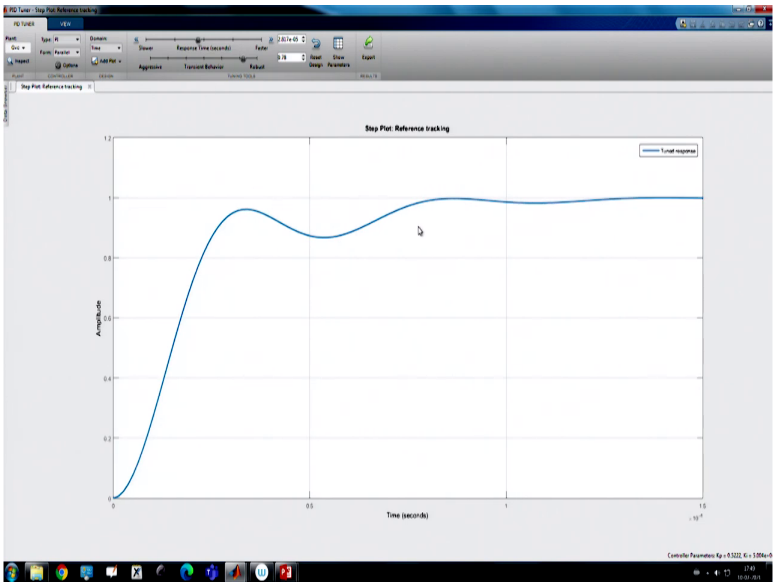
We want to make ok. So, we want to make ok.

(Refer Slide Time: 37:12)



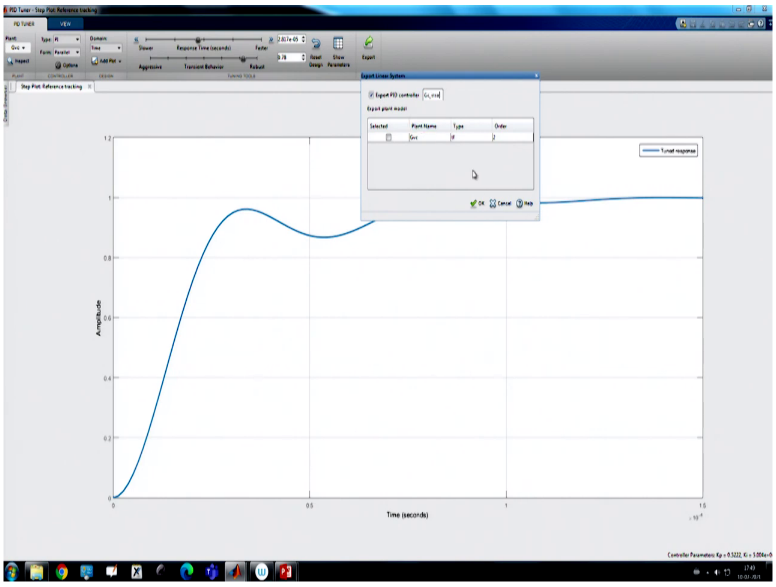
So, this is a design, we are try we are tuning this.

(Refer Slide Time: 37:15)



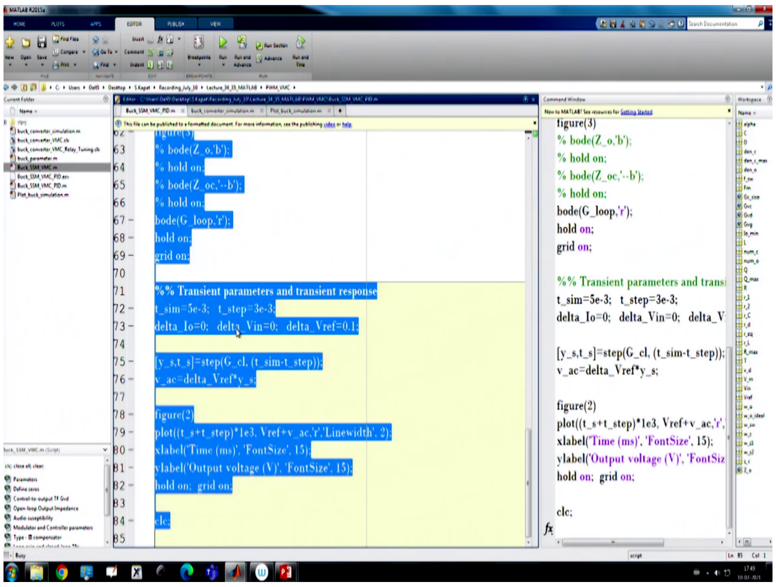
Because we are setting the so that we want to achieve the response. We want to achieve the steady state response near to this roughly 0.8; that means, 80 micro second. That means, in our case, it is like a 40 cycle like, because 2 micro second is my time period.

(Refer Slide Time: 37:38)



Now, we want to import this parameter that I have discussed. So, the parameter if you see this file I have called as Ge siso. So; that means, we need to copy it and go to this parameter and we just paste it ok, and we import it. Now, once we import now we want to run this rest of the file.

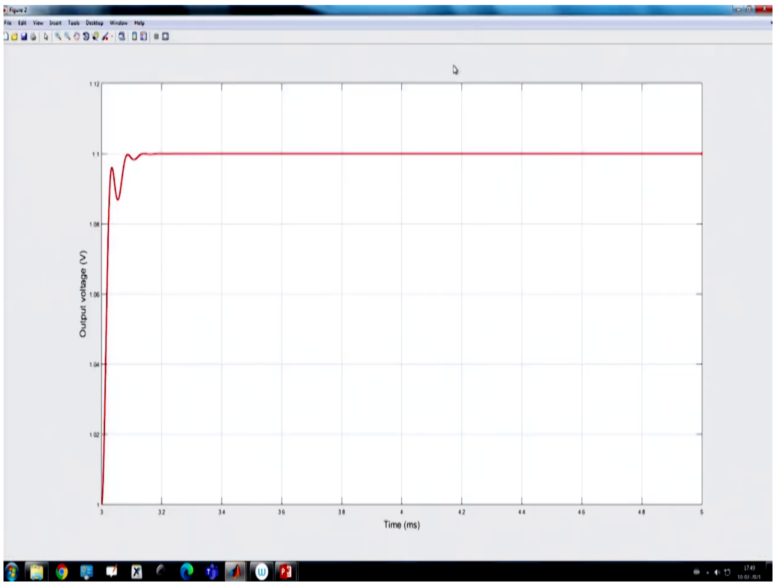
(Refer Slide Time: 37:57)



```
figure(3)
% bode(Z_o,'b');
% hold on;
% bode(Z_oc,'--b');
% hold on;
bode(G_loop,'r');
hold on;
grid on;

%% Transient parameters and transient response
t_sim=5e-3;   t_step=3e-3;
delta_Io=0;   delta_Vin=0;   delta_Vref=0.1;

[y_s,t_s]=step(G_cl, (t_sim-t_step));
v_ac=delta_Vref*y_s;

figure(2)
plot((t_s+t_step)*1e3, Vref+v_ac,'r','Linewidth', 2);
xlabel('Time (ms)', 'FontSize', 15);
ylabel('Output voltage (V)', 'FontSize', 15);
hold on;  grid on;

clc;
```

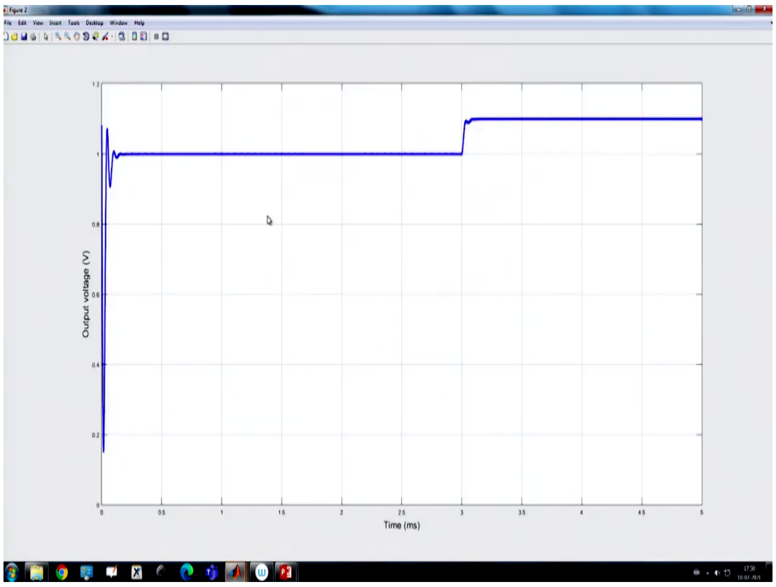
So, rest of the file I am running ok.

(Refer Slide Time: 38:01)



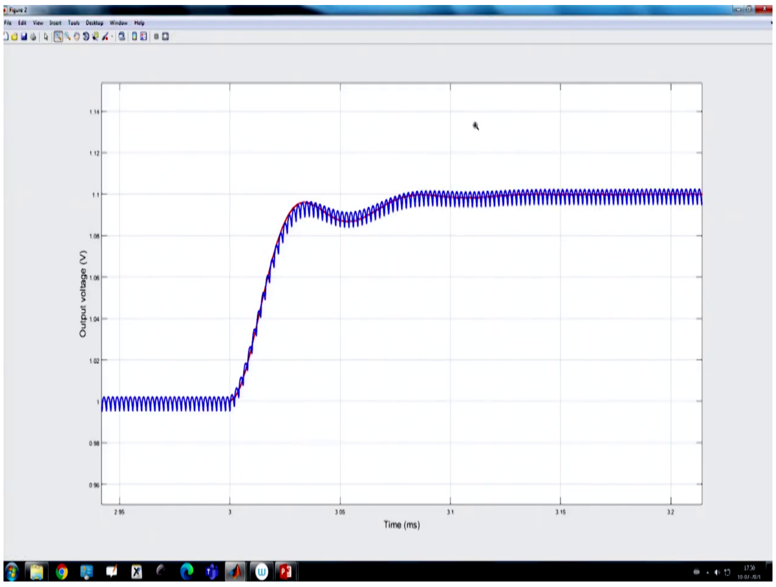
Now, we obtain this is the response of the controller, which we have designed using siso toolbox and we want to see and we want to match; now we want to run the converter ok. Because, we want to verify whether the converter, now converter, is loaded with this PID controller with the parameter, that you obtain from the siso toolbox at the PID toolbox ok.

(Refer Slide Time: 38:24)

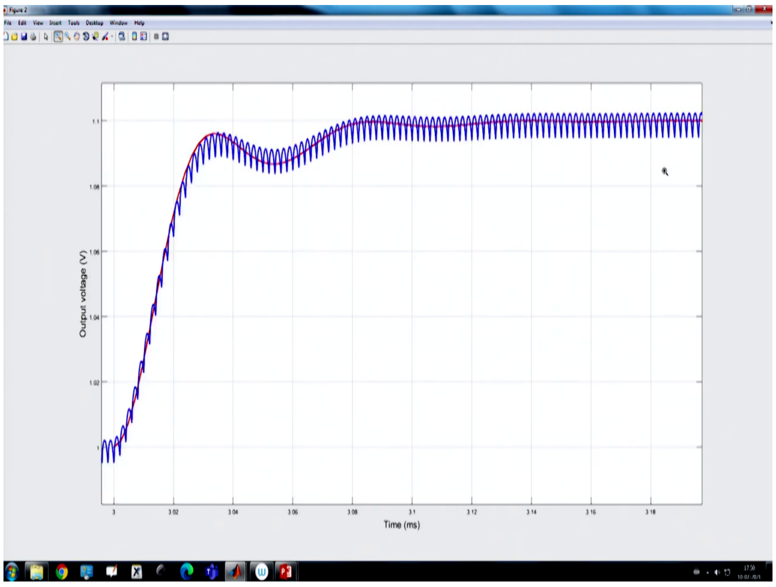


Let us run it and check whether ok. So, this is a response. I want to show that.

(Refer Slide Time: 38:28)

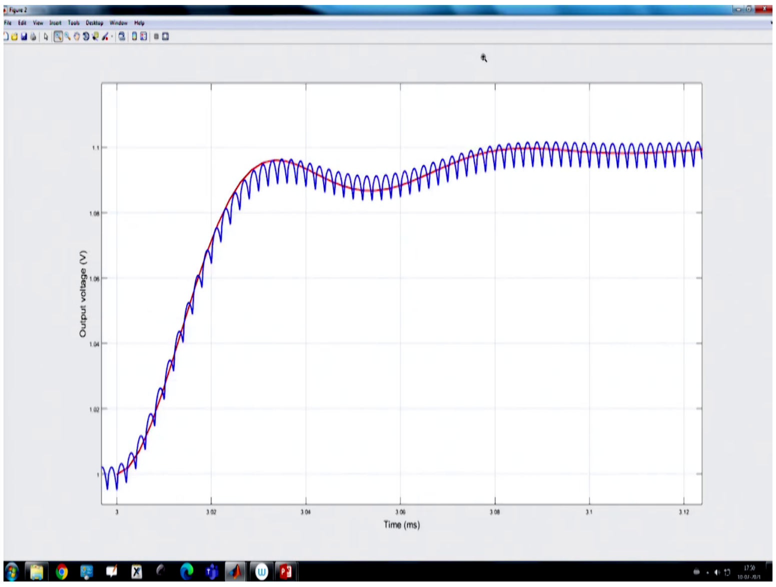


(Refer Slide Time: 38:30)



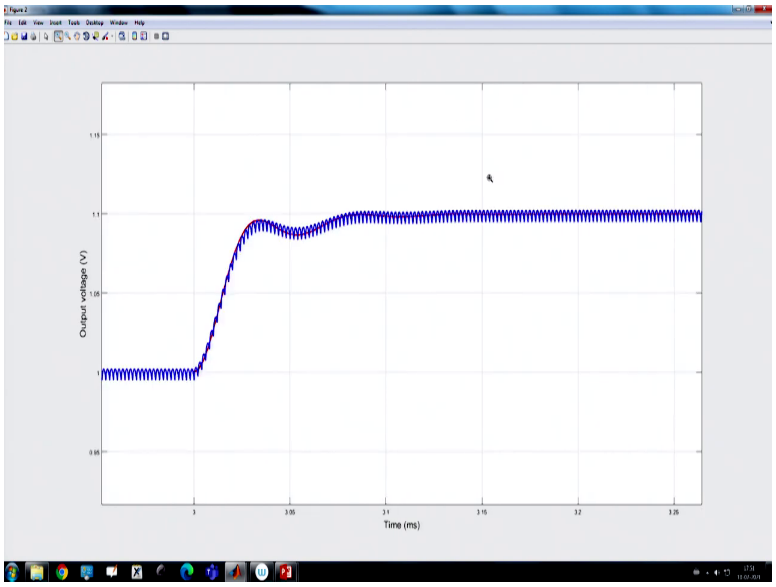
So, response looks matching reasonable with a reasonable accuracy at 3 millisecond, that we have applied a transient and you see we are reaching steady state near about this time.

(Refer Slide Time: 38:42)



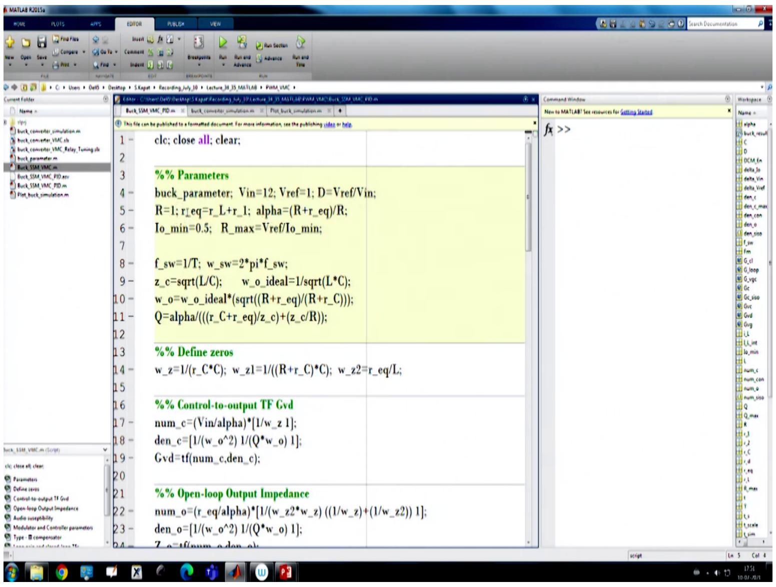
That means, if we reach here around 0.08; that means, in 80 cycle which we have set from that siso toolbox; that means, we said 80 micro second. Because, it is in millisecond so, it will be 0.08; that means, it is taking if a for a 1 10 cycle, 1, 2, 3, 4. So, sorry 20 cycle sorry it is a 2 micro second so, 2 micro second here there are 10 cycle, 20 cycle, 30 cycle, 40 so, 40 cycles. So, 80 micro second means we are talking about 40 cycles that we have already discussed. So, it is a steady state.

(Refer Slide Time: 39:21)



That means, this behavior is much better than what we achieved in the earlier using exact pole zero cancellation, because it was 1 degree-of-freedom control, but here we have designed this PID controller based on transient specification. We design at very high load condition, when the load resistance was very small.

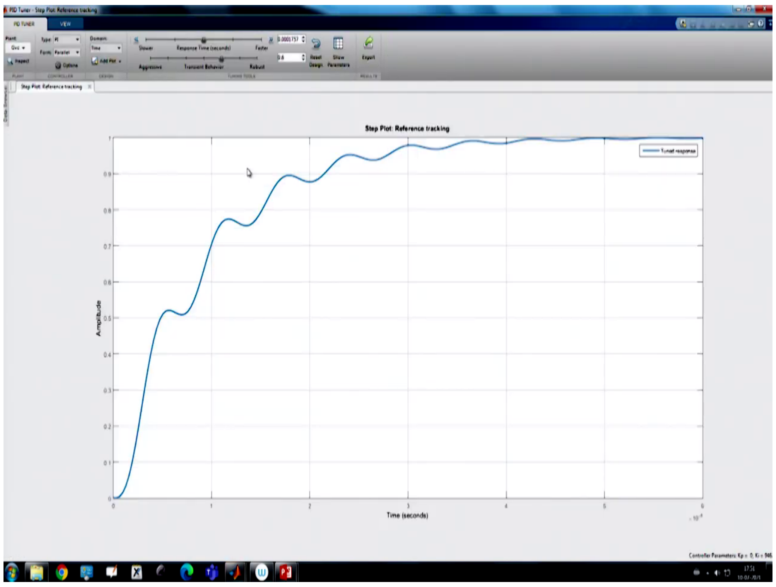
(Refer Slide Time: 39:42)



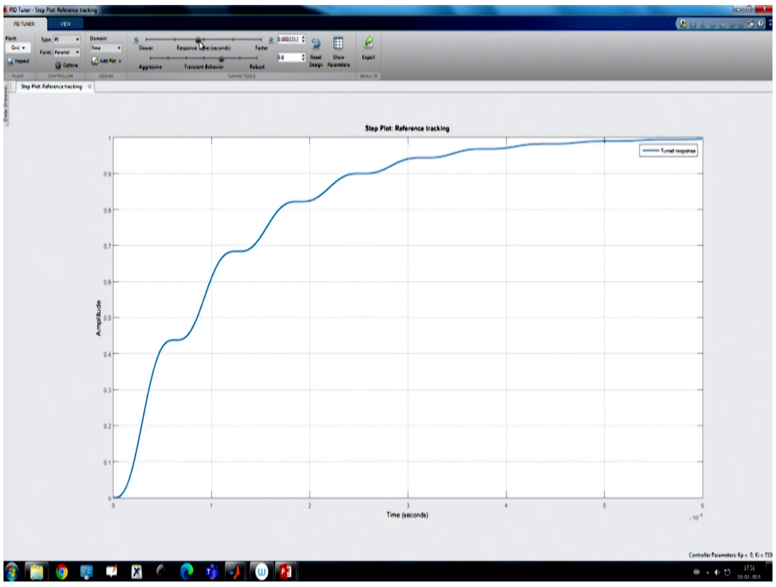
```
clc; close all; clear;

%% Parameters
buck_parameter;  Vin=12; Vref=1; D=Vref/Vin;
R=1; r_eq=r_L+r_1;  alpha=(R+r_eq)/R;
Io_min=0.5;   R_max=Vref/Io_min;

f_sw=1/T;  w_sw=2*pi*f_sw;
z_c=sqrt(L/C);      w_o_ideal=1/sqrt(L*C);
w_o=w_o_ideal*(sqrt((R+r_eq)/(R+r_C)));
Q=alpha/(((r_C+r_eq)/z_c)+(z_c/R));

%% Define zeros
w_z=1/(r_C*C);  w_z1=1/((R+r_C)*C);  w_z2=r_eq/L;

%% Control-to-output TF Gvd
num_c=(Vin/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvd=tf(num_c,den_c);

%% Open-loop Output Impedance
num_o=(r_eq/alpha)*[1/(w_z2*w_z) ((1/w_z)+(1/w_z2)) 1];
den_o=[1/(w_o^2) 1/(Q*w_o) 1];
```

Now, suppose we increase the load resistance. Now the load current is roughly 1 ampere, earlier it was 20 ampere. So, we have a very good damping. Now, it is with lower damping and we want to run this we want to design the converter.

(Refer Slide Time: 39:53)

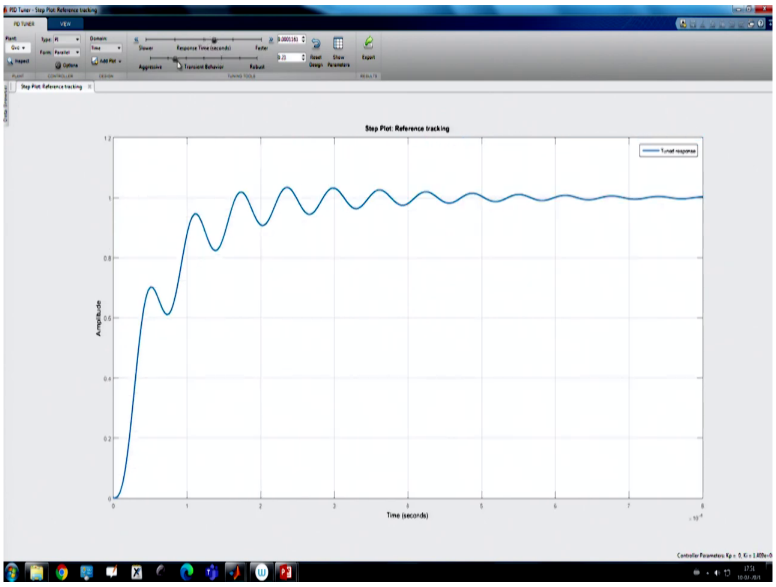


(Refer Slide Time: 39:56)



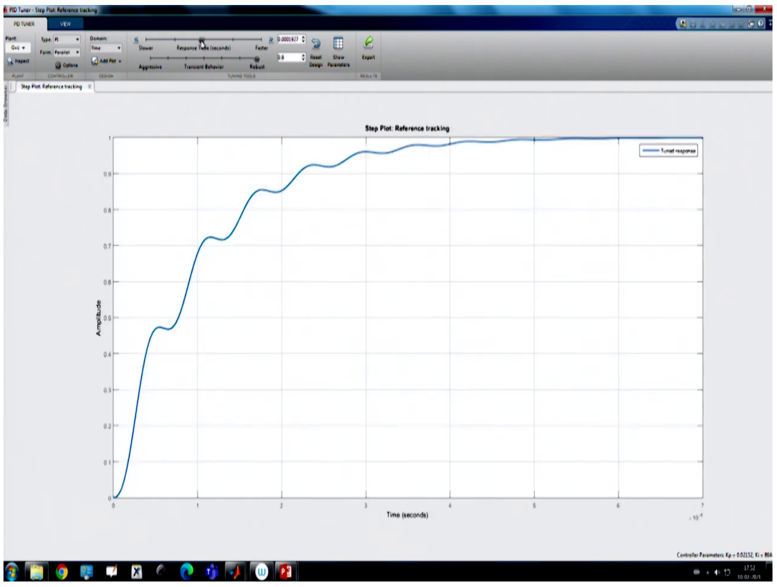
And, you see, in this case, you have real difficulty with the design converter design.

(Refer Slide Time: 39:58)

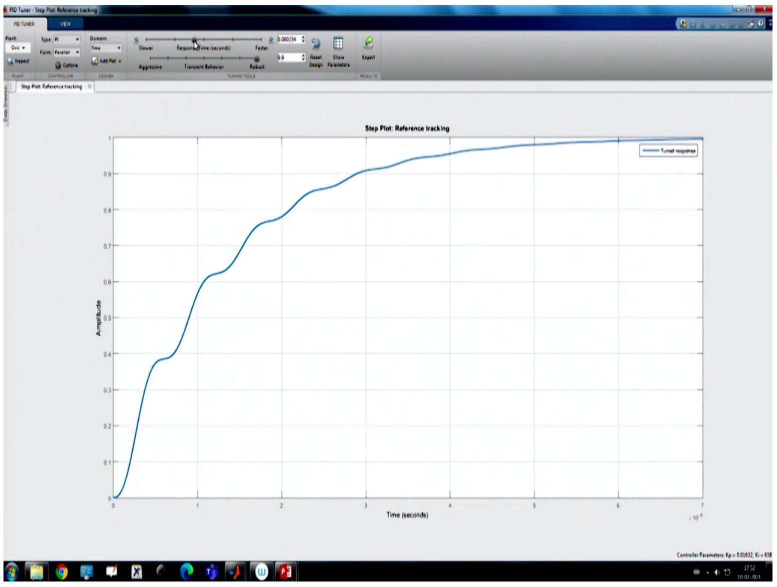


So, if we want to like you know, make that. We have a little choice over this.

(Refer Slide Time: 40:04)

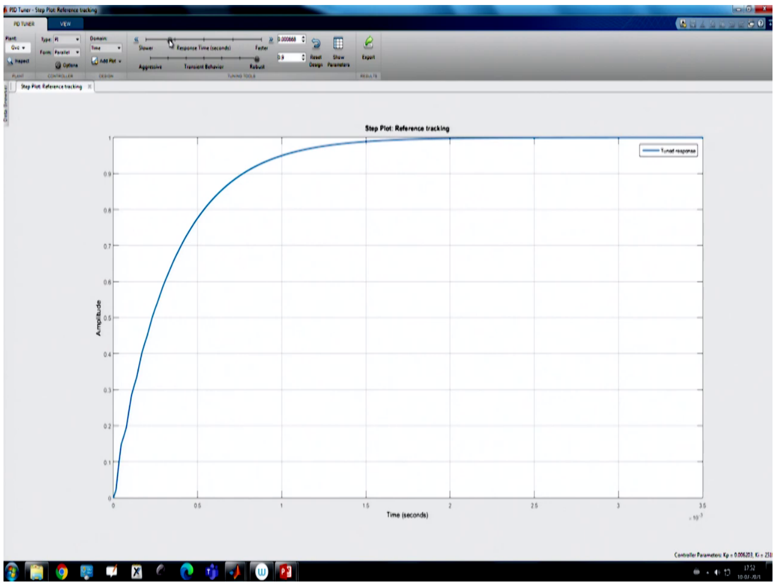


(Refer Slide Time: 40:07)



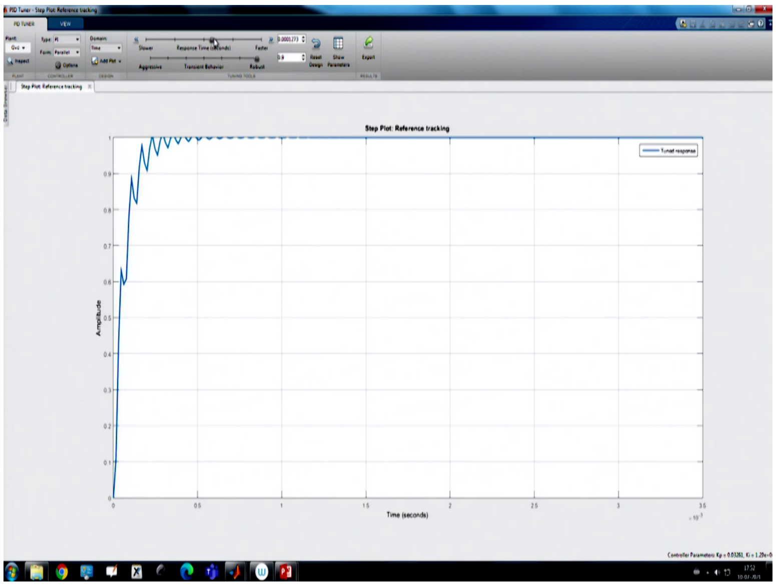
If you want to speed up the response at you know.

(Refer Slide Time: 40:09)



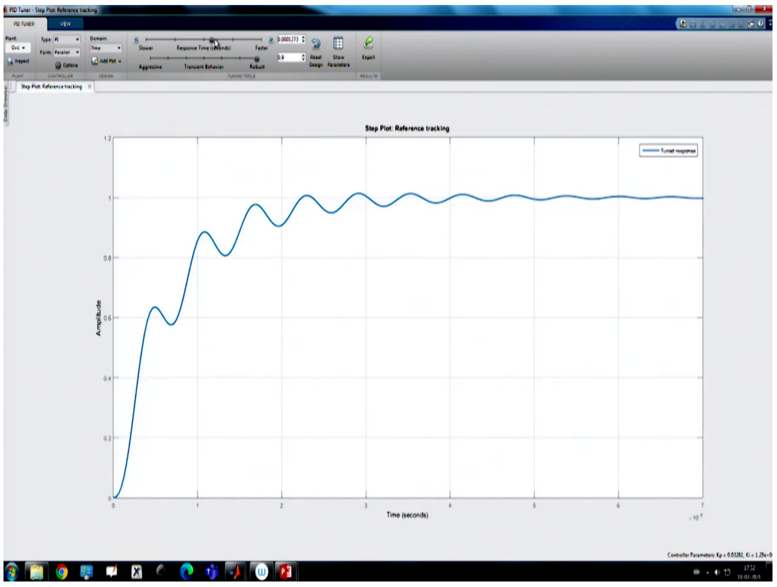
So, this is the response speed and we want to see whether, so, this is like you know now we are trying to achieve in this is in millisecond in 2 milliseconds. So, we are making it slower, if we make it slow; that means, if we try to make it fast; that means, you know.

(Refer Slide Time: 40:24)



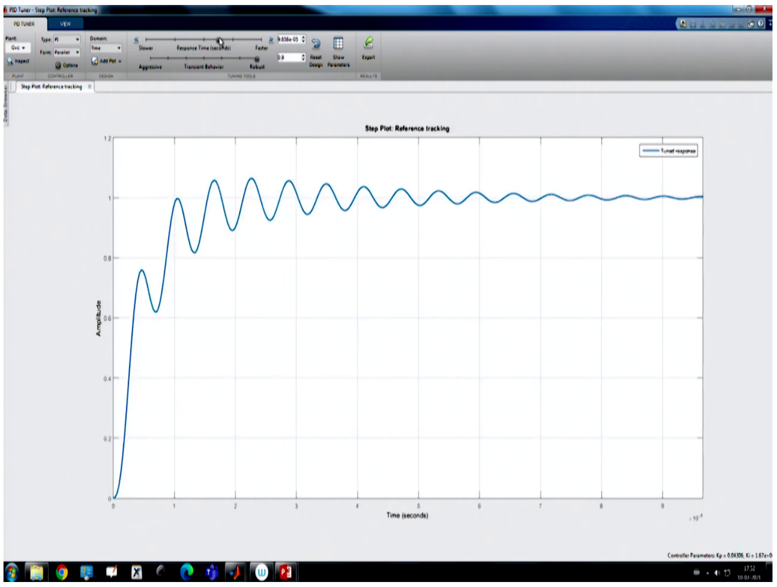
If we try to make it fast you know like earlier case.

(Refer Slide Time: 40:27)

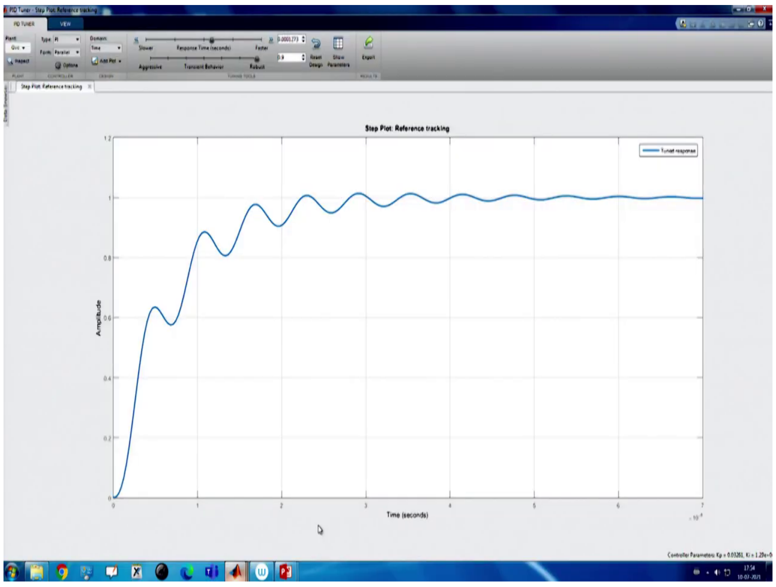


We want to achieve much faster.

(Refer Slide Time: 40:38)

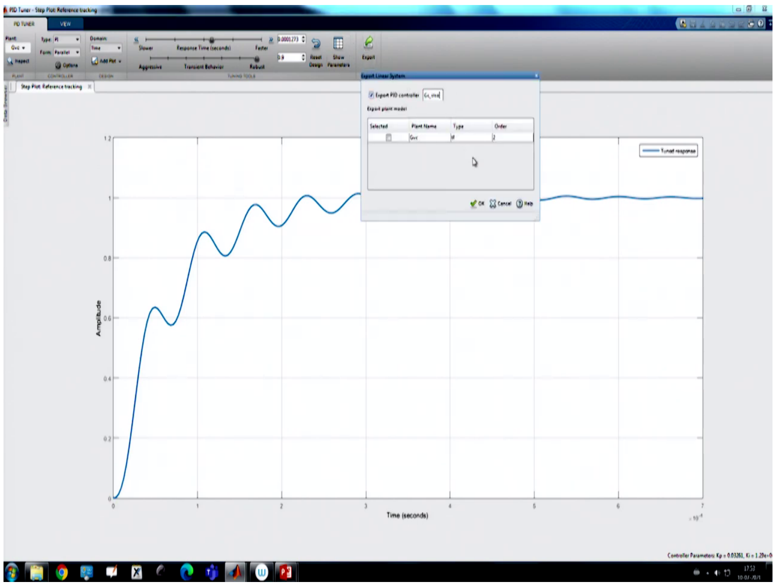


(Refer Slide Time: 40:39)



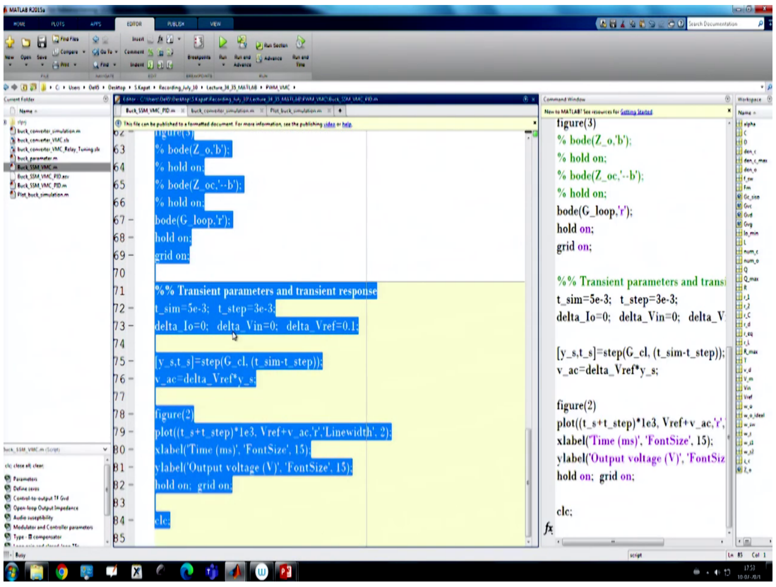
Then, there is a penalty in ok. You want to achieve and to make it fast ok. So, now we want to again you know import the controller parameter. So, this is a Ge siso tool. Let us go to the controller and sorry the GUI tool.

(Refer Slide Time: 41:04)



And, in this tool, we want to import the controller parameter. We have imported the controller parameter now we want to run it. So, we want to run it ok.

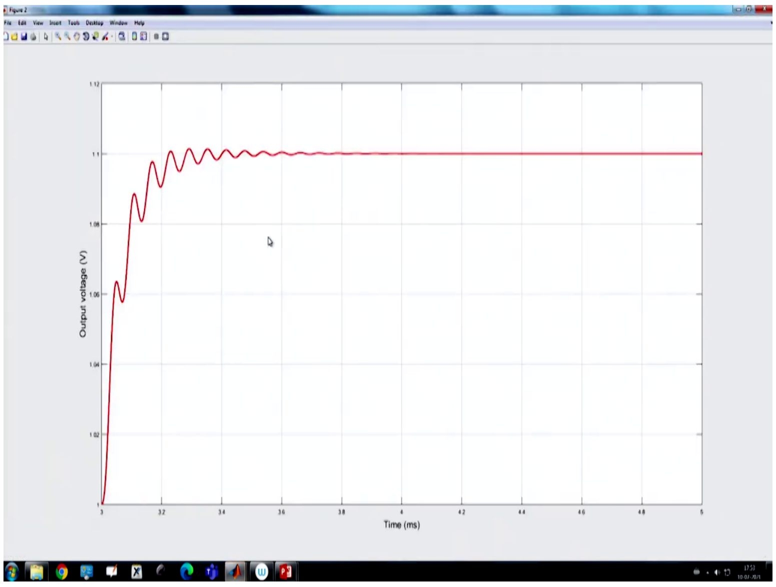
(Refer Slide Time: 41:16)



```
figure(3)
% bode(Z_o,'b');
% hold on;
% bode(Z_oc,'--b');
% hold on;
bode(G_loop,'r');
hold on;
grid on;

%% Transient parameters and transient response
t_sim=5e-3;  t_step=3e-3;
delta_Io=0;  delta_Vin=0;  delta_Vref=0.1;

[y_s,t_s]=step(G_cl, (t_sim-t_step));
v_ac=delta_Vref*y_s;

figure(2)
plot((t_s+t_step)*1e3, Vref+v_ac,'r','Linewidth', 2);
xlabel('Time (ms)', 'FontSize', 15);
ylabel('Output voltage (V)', 'FontSize', 15);
hold on;  grid on;

clc;
```

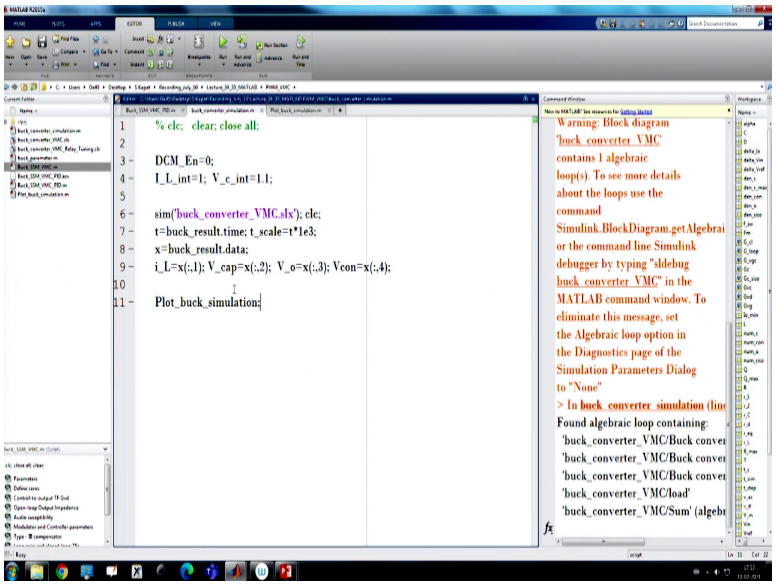
```
figure(3)
% bode(Z_o,'b');
% hold on;
% bode(Z_oc,'--b');
% hold on;
bode(G_loop,'r');
hold on;
grid on;

%% Transient parameters and transi
t_sim=5e-3;  t_step=3e-3;
delta_Io=0;  delta_Vin=0;  delta_V

[y_s,t_s]=step(G_cl, (t_sim-t_step));
v_ac=delta_Vref*y_s;

figure(2)
plot((t_s+t_step)*1e3, Vref+v_ac,'r',
xlabel('Time (ms)', 'FontSize', 15);
ylabel('Output voltage (V)', 'FontSize
hold on;  grid on;

clc;
```

(Refer Slide Time: 41:18)



So, you can see this is a response of the system, because the response is system has a port damping and the PID controller is designed.

(Refer Slide Time: 41:29)

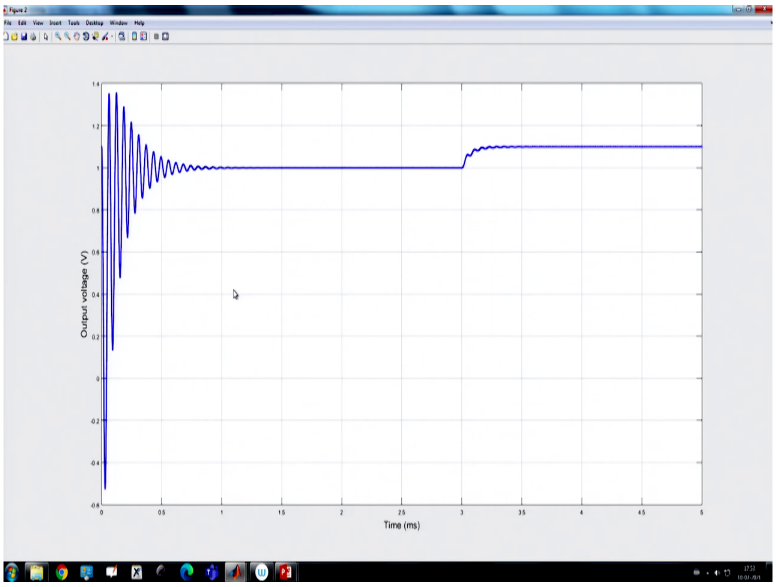


```
% clc;   clear; close all;

DCM_En=0;
I_L_int=1;  V_c_int=1.1;

sim('buck_converter_VMC.slx'); clc;
t=buck_result.time; t_scale=t*1e3;
x=buck_result.data;
i_L=x(:,1); V_cap=x(:,2);  V_o=x(:,3); Vcon=x(:,4);

Plot_buck_simulation;
```

```
Warning: Block diagram
'buck_converter_VMC'
contains 1 algebraic
loop(s). To see more details
about the loops use the
command
Simulink.BlockDiagram.getAlgebrai
or the command line Simulink
debugger by typing "sldebug
buck_converter_VMC" in the
MATLAB command window. To
eliminate this message, set
the Algebraic loop option in
the Diagnostics page of the
Simulation Parameters Dialog
to "None"
> In buck_converter_simulation (line
Found algebraic loop containing:
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/load'
 'buck_converter_VMC/Sum' (algebr
```

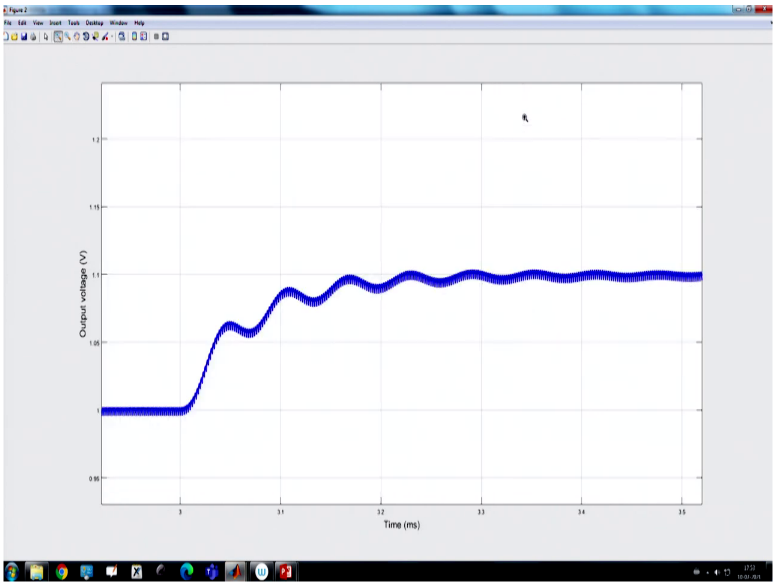
Now, we want to run the actual simulation and see what happens?

(Refer Slide Time: 41:33)



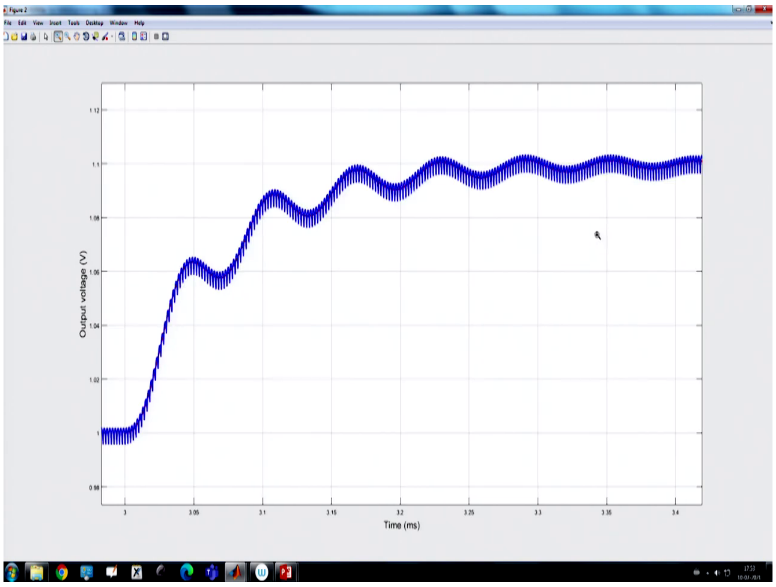
So, this is a we want to match the response of the converter.

(Refer Slide Time: 41:36)

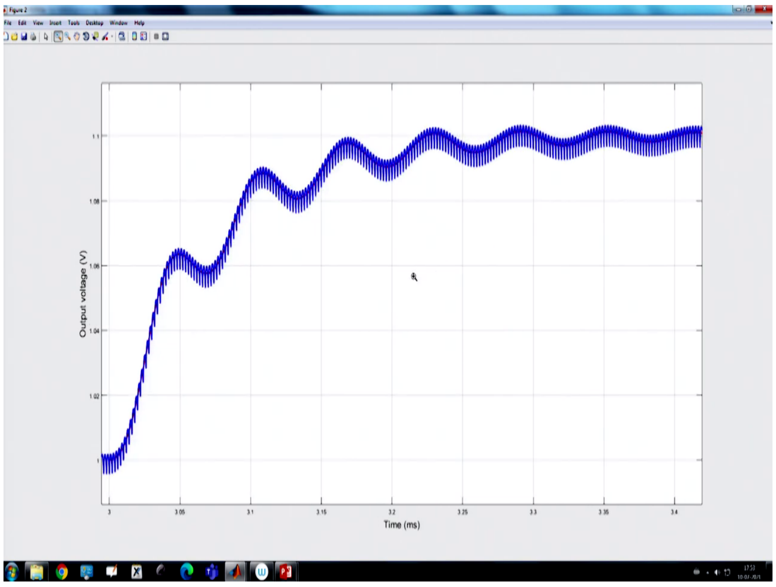


And, you see, the converter responses match quite nicely.

(Refer Slide Time: 41:38)

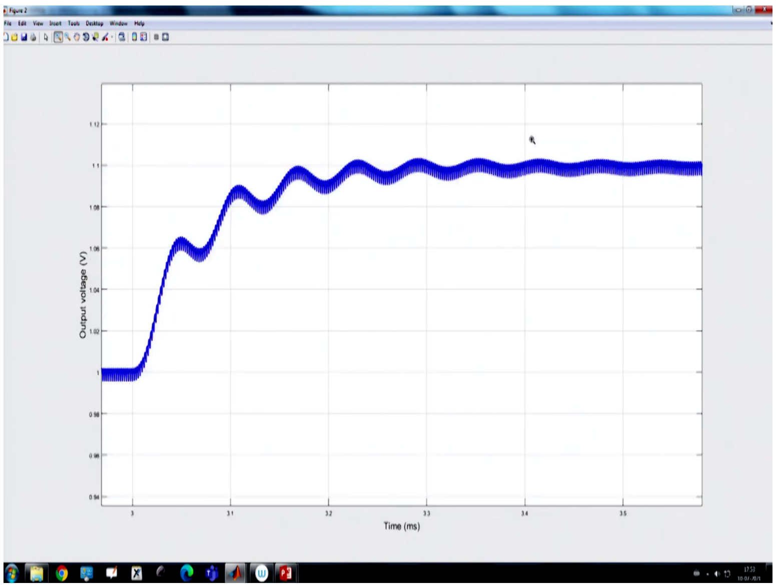


(Refer Slide Time: 41:41)



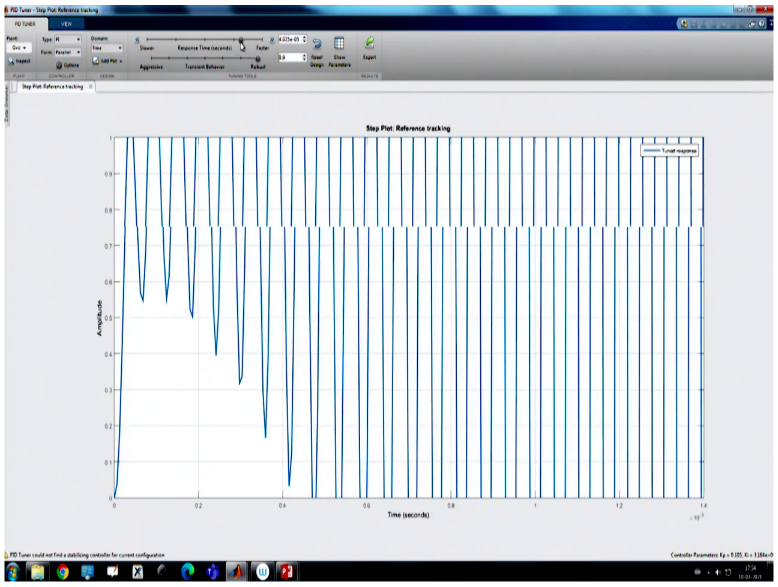
But, this response is not satisfactory enough.

(Refer Slide Time: 41:49)



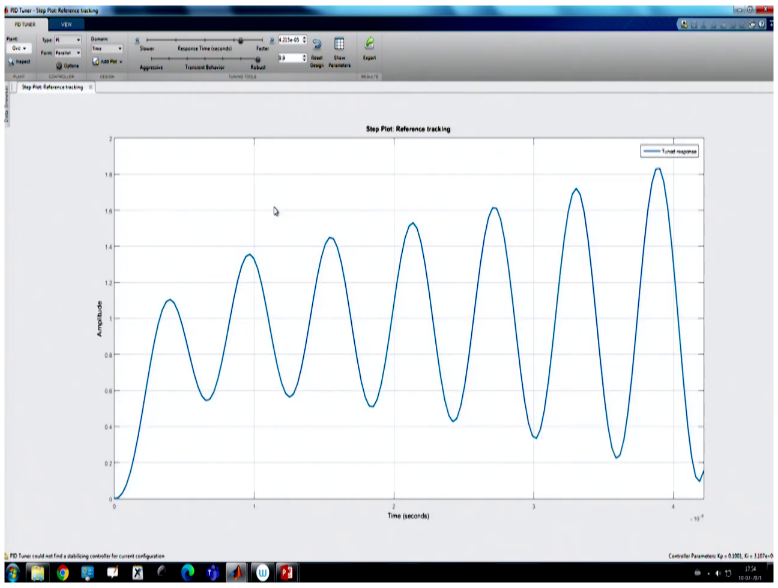
Because you know, it is not good enough, because you have to sacrifice the settling time and if you try to increase the speed of the response, the system will simply become unstable. Because, you see, at 3 millisecond we have applied load step, and it is reaching a steady state at 0.3 or roughly and 0.4. That means, it is taking around 200 cycles, which is pretty large in terms of transient response.

(Refer Slide Time: 42:18)

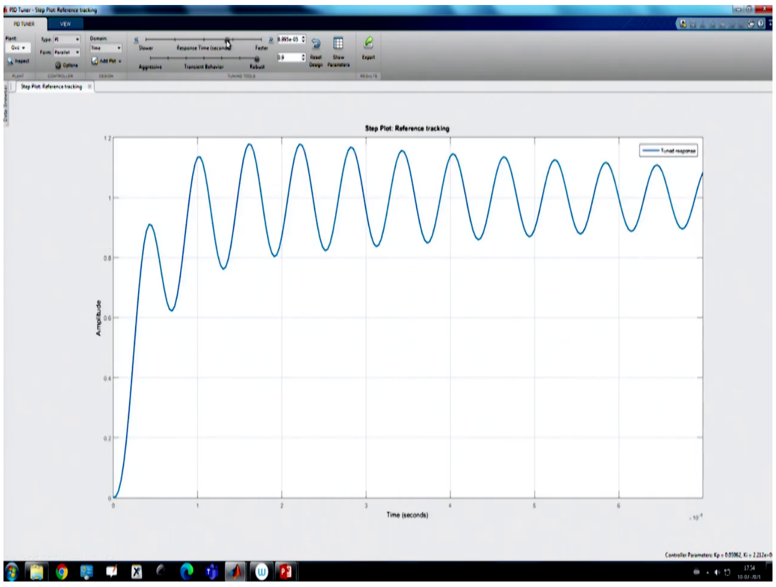


And, if we want to improve this response by because if you see, if you want to make this response faster; that means, if you want to speed up, it will simply become unstable.

(Refer Slide Time: 42:20)

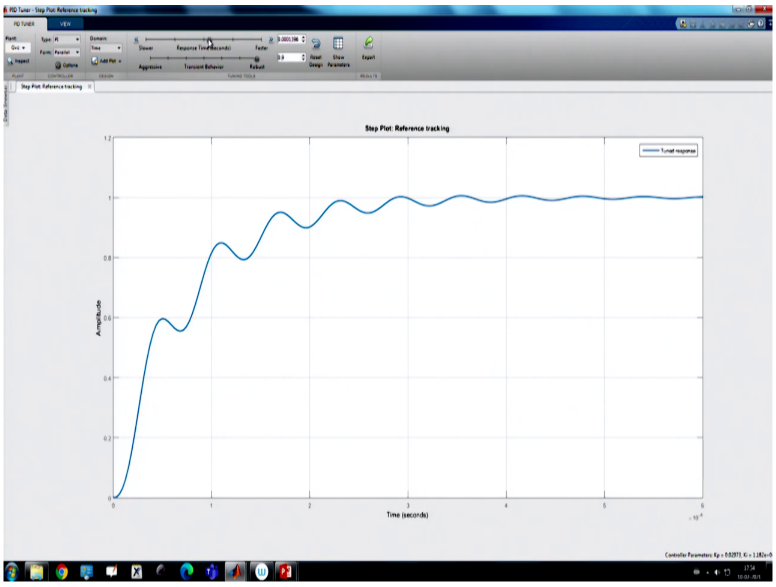


(Refer Slide Time: 42:22)



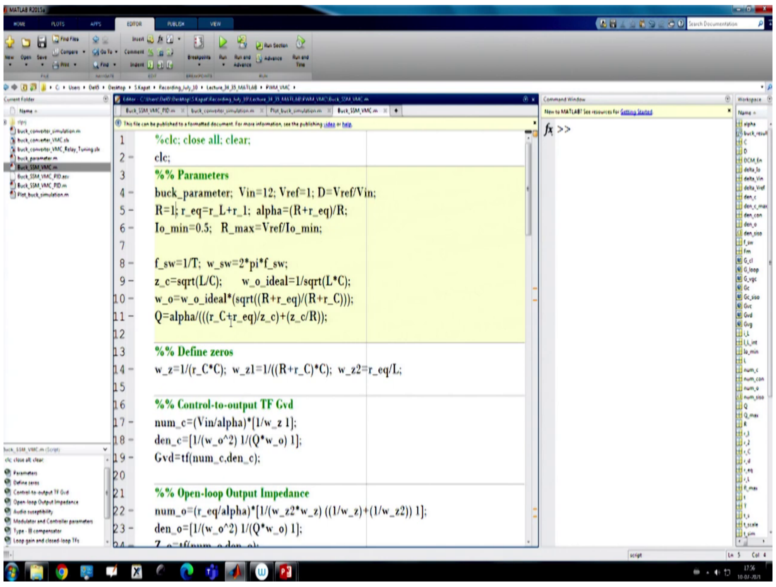
Because of the similar response we can obtain.

(Refer Slide Time: 42:26)



So, we cannot improve further, because then it will become unstable, because it has a poor phase margin ok. So, we have we got a limited response due to this. Now, we want to show you that under this condition, if we go for another compensator.

(Refer Slide Time: 42:44)



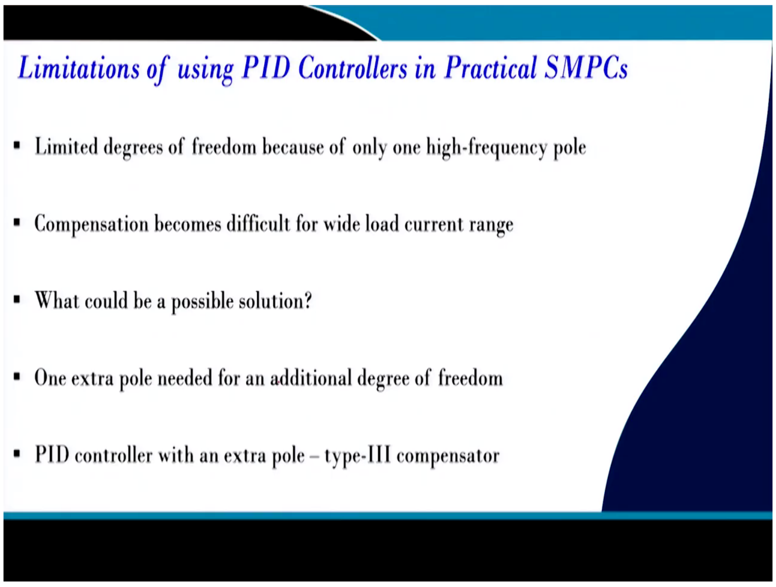
```
%clc; close all; clear;
clc;
%% Parameters
buck_parameter;  Vin=12; Vref=1; D=Vref/Vin;
R=1; r_eq=r_L+r_1;  alpha=(R+r_eq)/R;
Io_min=0.5;   R_max=Vref/Io_min;

f_sw=1/T;  w_sw=2*pi*f_sw;
z_c=sqrt(L/C);      w_o_ideal=1/sqrt(L*C);
w_o=w_o_ideal*(sqrt((R+r_eq)/(R+r_C)));
Q=alpha/(((r_C+r_eq)/z_c)+(z_c/R));

%% Define zeros
w_z=1/(r_C*C);  w_z1=1/((R+r_C)*C);  w_z2=r_eq/L;

%% Control-to-output TF Gvd
num_c=(Vin/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvd=tf(num_c,den_c);

%% Open-loop Output Impedance
num_o=(r_eq/alpha)*[1/(w_z2*w_z) ((1/w_z)+(1/w_z2)) 1];
den_o=[1/(w_o^2) 1/(Q*w_o) 1];
```

That means, let us go for another compensator where the same parameter condition at 1 ohm we want to design, because that we will discuss. So, PID controller even we design with cancellation pole zero cancellation, we design with transient specification.

And, in the first case, we found the model validity even at a lower cutoff frequency. In the second case, we found the model is very good is valid, but we have a very slower transient performance. As well as the oscillator behavior at particularly what is the problem, because PID controller has a limited flexibility.

Now, we are going to consider another converter which is like not near which is name which is called as type-III compensator. We want to add 1 filter ok.

(Refer Slide Time: 43:32)



## Limitations of using PID Controllers in Practical SMPCs

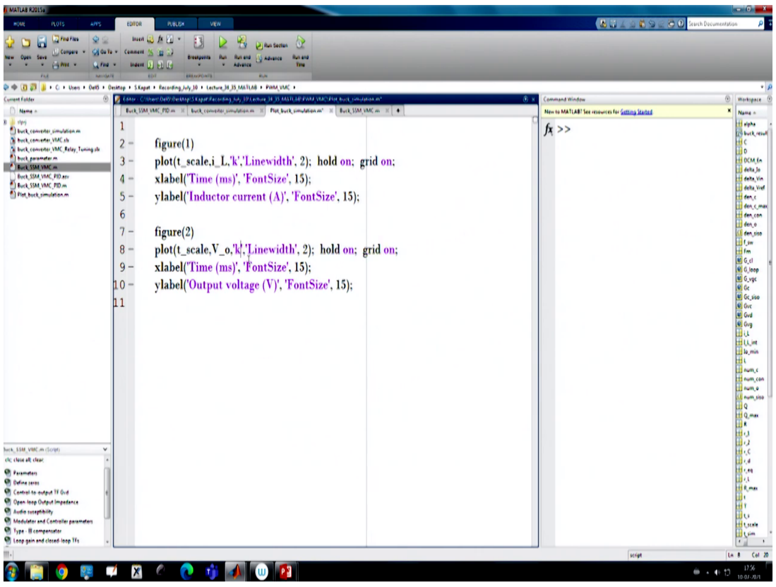
- Limited degrees of freedom because of only one high-frequency pole
- Compensation becomes difficult for wide load current range
- What could be a possible solution?
- One extra pole needed for an additional degree of freedom
- PID controller with an extra pole – type-III compensator

Before that, let us discuss, so, we run this and we found limited degree of freedom because of only 1 high frequency pole, compensation become difficult for wide load current range, what could be a possible solution. So, we need to consider 1 extra pole for an additional degree of freedom.

So, if we add one extra pole with a practical PID controller then it becomes a type III compensator, which has two zeros and three poles: one pole is at the origin, one is a derivative pole, which can cancel the ESR. And, the third pole is the one which we have an additional degree of freedom. And, that we will discuss in the next class the design method.

But, this class we want to see what happens? If we design this using type III compensator. Now, in this method we want to design and we want to make sure that we use a different plot color ok.

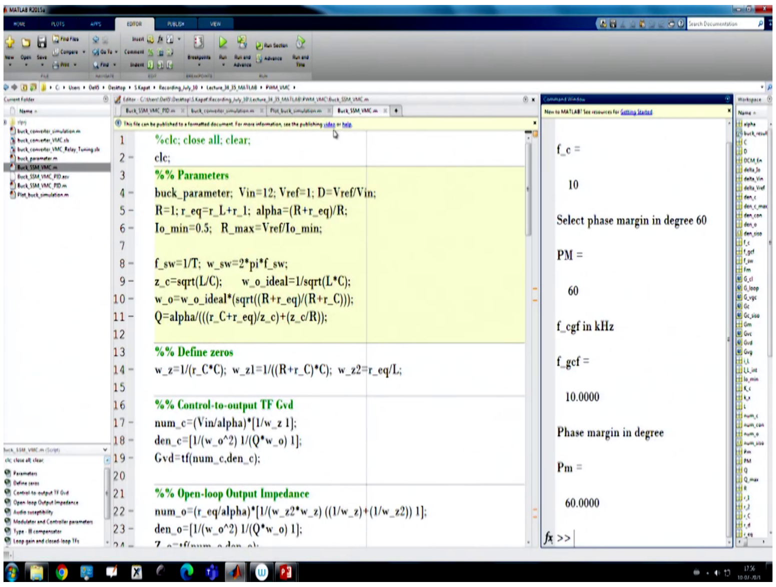
(Refer Slide Time: 44:26)



```
figure(1)
plot(t_scale,i_L,'k','Linewidth', 2);  hold on;  grid on;
xlabel('Time (ms)', 'FontSize', 15);
ylabel('Inductor current (A)', 'FontSize', 15);

figure(2)
plot(t_scale,V_o,'k','Linewidth', 2);  hold on;  grid on;
xlabel('Time (ms)', 'FontSize', 15);
ylabel('Output voltage (V)', 'FontSize', 15);
```

So, you want to use you know black color plot ok.

(Refer Slide Time: 44:31)

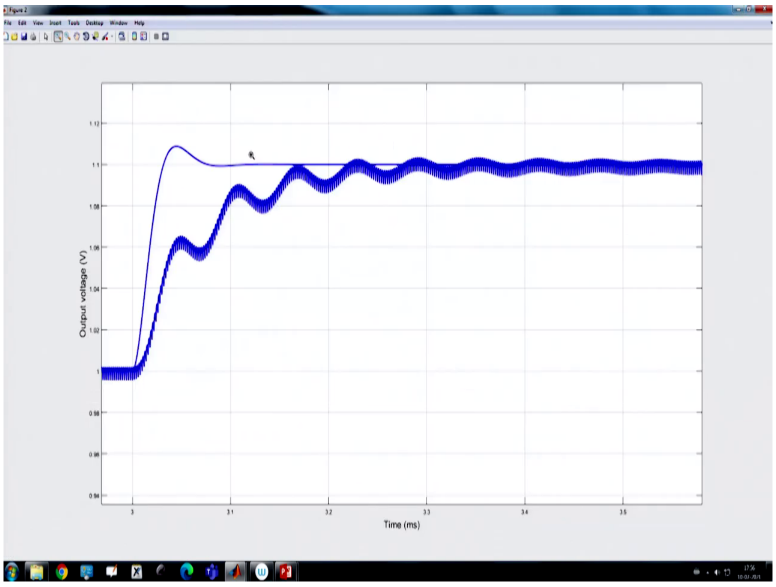


```
%clc; close all; clear;
clc;
%% Parameters
buck_parameter;  Vin=12; Vref=1; D=Vref/Vin;
R=1; r_eq=r_L+r_1;  alpha=(R+r_eq)/R;
Io_min=0.5;   R_max=Vref/Io_min;

f_sw=1/T;  w_sw=2*pi*f_sw;
z_c=sqrt(L/C);      w_o_ideal=1/sqrt(L*C);
w_o=w_o_ideal*(sqrt((R+r_eq)/(R+r_C)));
Q=alpha/(((r_C+r_eq)/z_c)+(z_c/R));

%% Define zeros
w_z=1/(r_C*C);  w_z1=1/((R+r_C)*C);  w_z2=r_eq/L;

%% Control-to-output TF Gvd
num_c=(Vin/alpha)*[1/w_z 1];
den_c=[1/(w_o^2) 1/(Q*w_o) 1];
Gvd=tf(num_c,den_c);

%% Open-loop Output Impedance
num_o=(r_eq/alpha)*[1/(w_z2*w_z) ((1/w_z)+(1/w_z2)) 1];
den_o=[1/(w_o^2) 1/(Q*w_o) 1];
```

```
f_c =

  10

Select phase margin in degree 60

PM =

  60

f_cgf in kHz

f_gcf =

  10.0000

Phase margin in degree

Pm =

  60.0000

fx >>
```

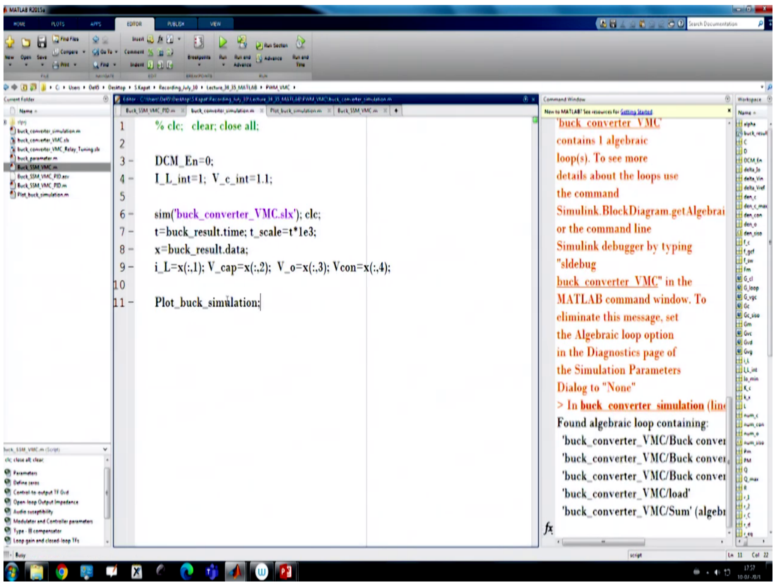
And, in this technique I have designed such that it will ask for bandwidth. Since it your we are operating at a light load. So, we have to careful about the phase margin right. So, let us say we talked about like a 20 kilohertz bandwidth ok or even maybe 10 kilohertz bandwidth and phase margin we want to set in 60 degrees ok.

(Refer Slide Time: 44:53)



So, this is a response which is coming using small-signal model and now we want to verify using our actual simulation ok.

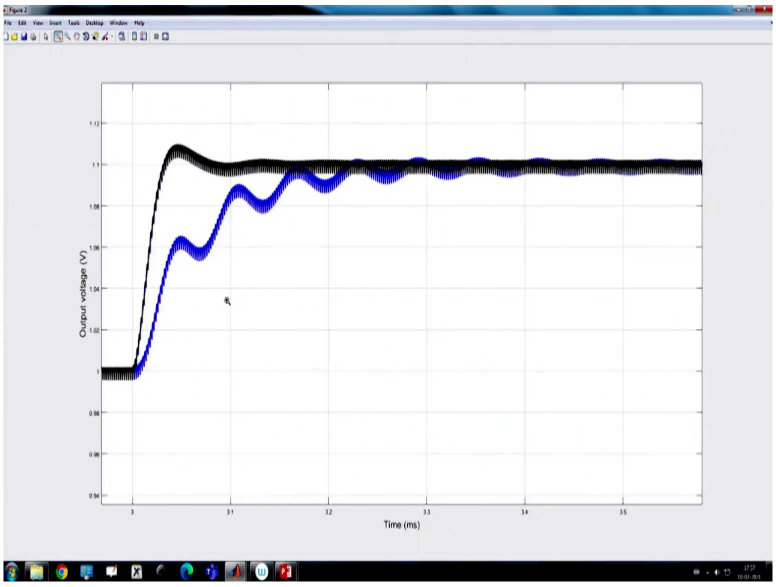
(Refer Slide Time: 45:05)



```
% clc;   clear; close all;

DCM_En=0;
I_L_int=1;  V_c_int=1.1;

sim('buck_converter_VMC.slx'); clc;
t=buck_result.time; t_scale=t*1e3;
x=buck_result.data;
i_L=x(:,1); V_cap=x(:,2);  V_o=x(:,3); Vcon=x(:,4);

Plot_buck_simulation;
```

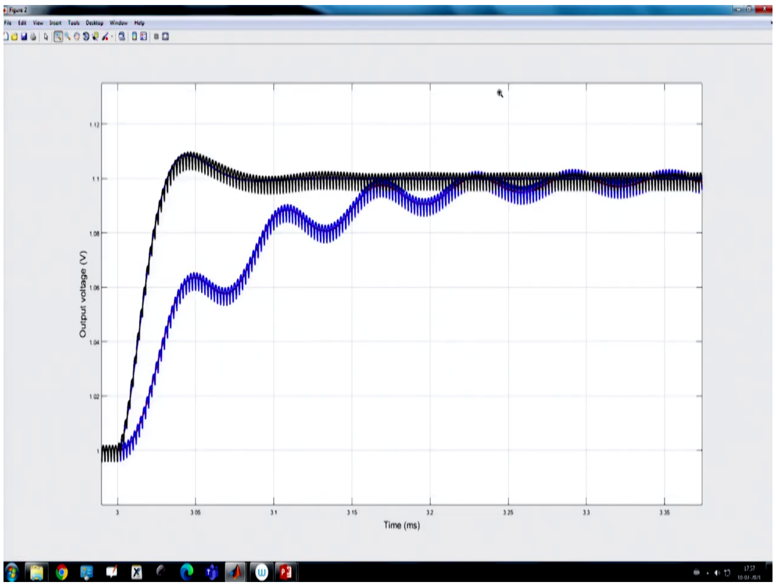
```
'buck_converter_VMC'
contains 1 algebraic
loop(s). To see more
details about the loops use
the command
Simulink.BlockDiagram.getAlgebrai
or the command line
Simulink debugger by typing
"sldebug
buck_converter_VMC" in the
MATLAB command window. To
eliminate this message, set
the Algebraic loop option
in the Diagnostics page of
the Simulation Parameters
Dialog to "None"
> In buck_converter_simulation (line
Found algebraic loop containing:
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/Buck conver
 'buck_converter_VMC/load'
 'buck_converter_VMC/Sum' (algebr
```

(Refer Slide Time: 45:08)



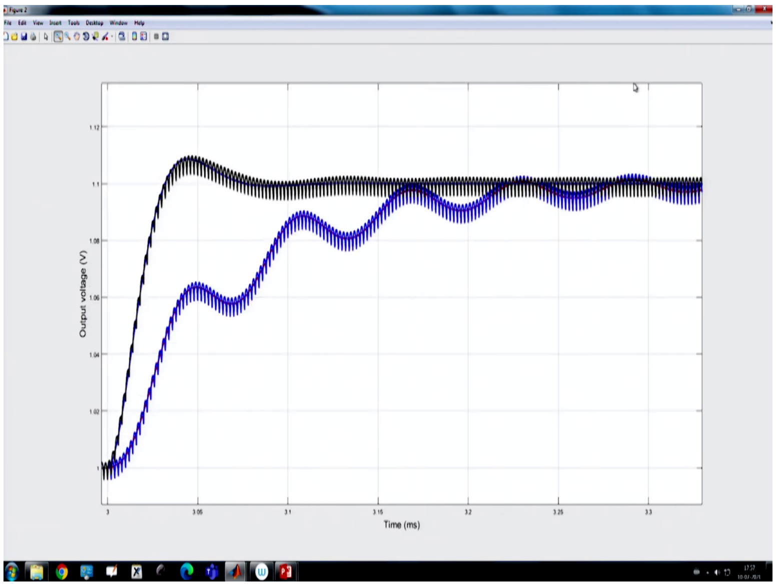
So, this is a type III compensator now.

(Refer Slide Time: 45:12)



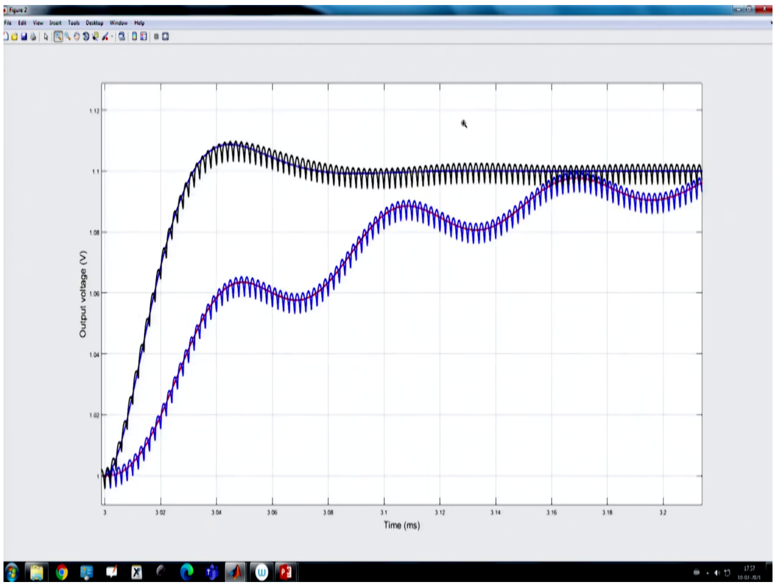
And, you see using type III compensator; we can speed up the response significantly.

(Refer Slide Time: 45:18)



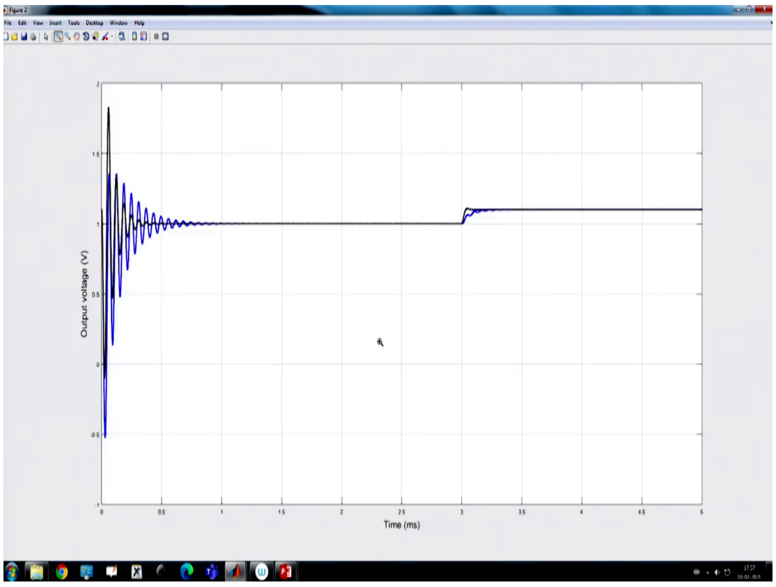
Because, at the same operating condition, the response can be drastically improved.

(Refer Slide Time: 45:21)



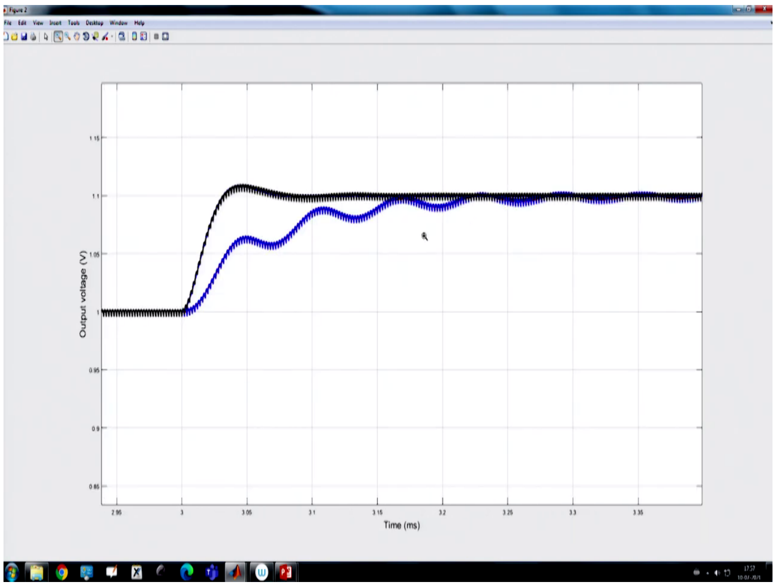
Because we can achieve settling time even in 40 you know switching cycle, which we did it in high load conditions.

(Refer Slide Time: 45:31)



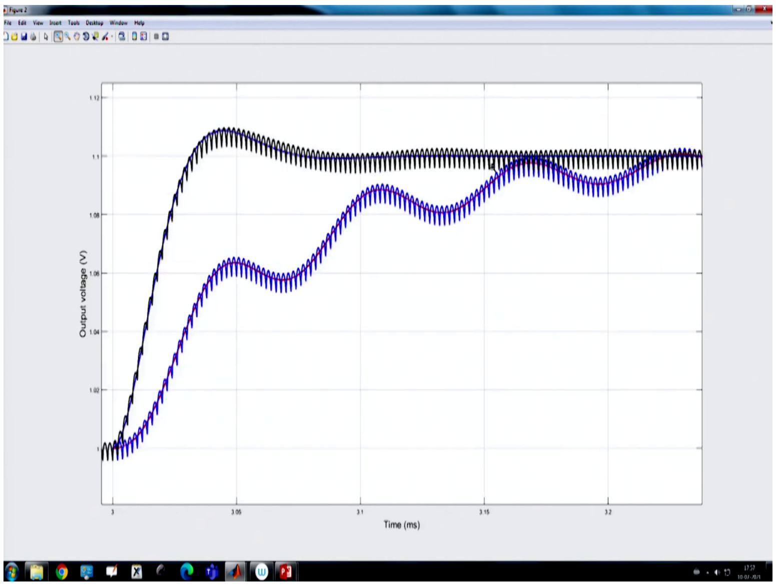
So, in light load we can well damped the system.

(Refer Slide Time: 45:32)



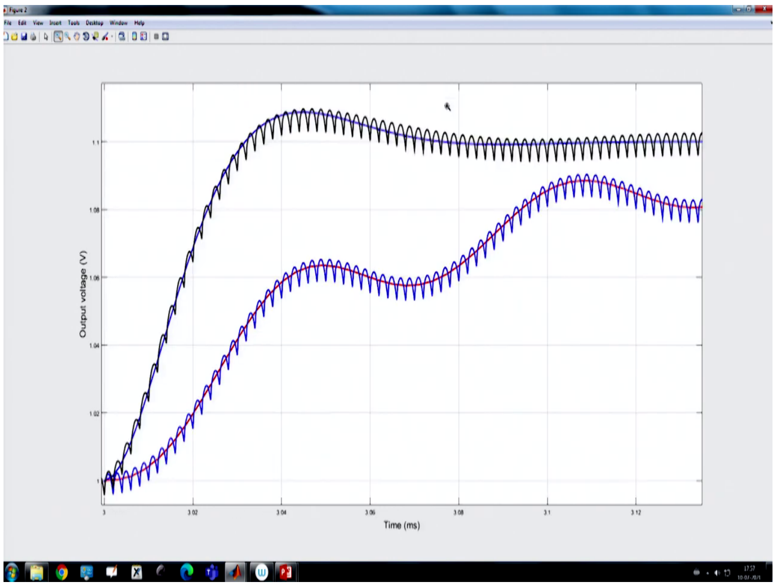
And, this is the matching response of the small-signal model.

(Refer Slide Time: 45:35)



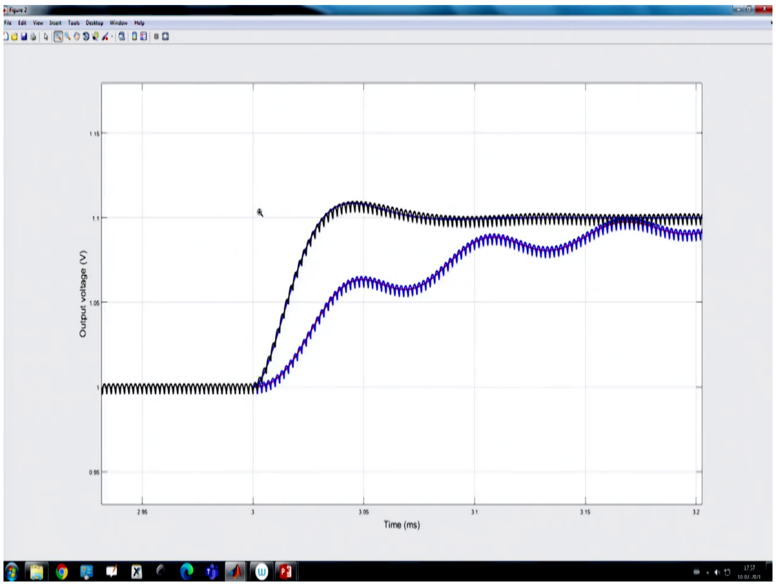
And, the actual switch simulation.

(Refer Slide Time: 45:38)



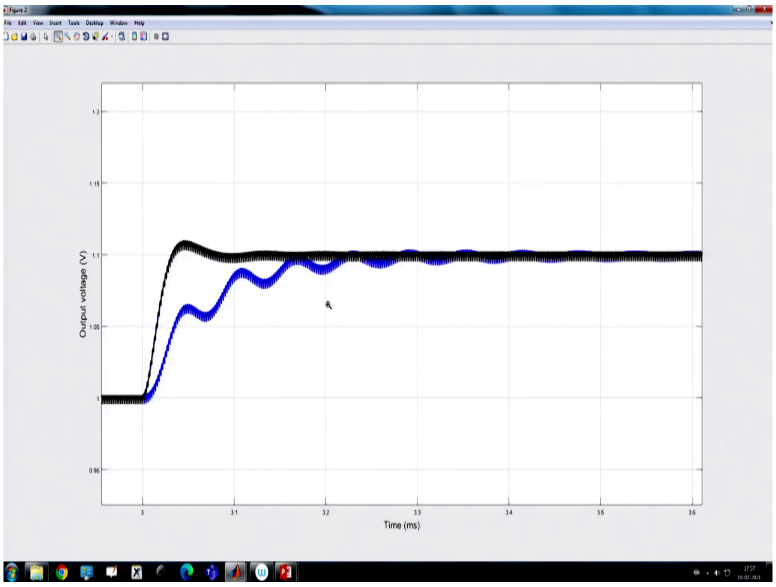
So, this is the second case with the type III compensator, they are matching quite nicely.

(Refer Slide Time: 45:43)

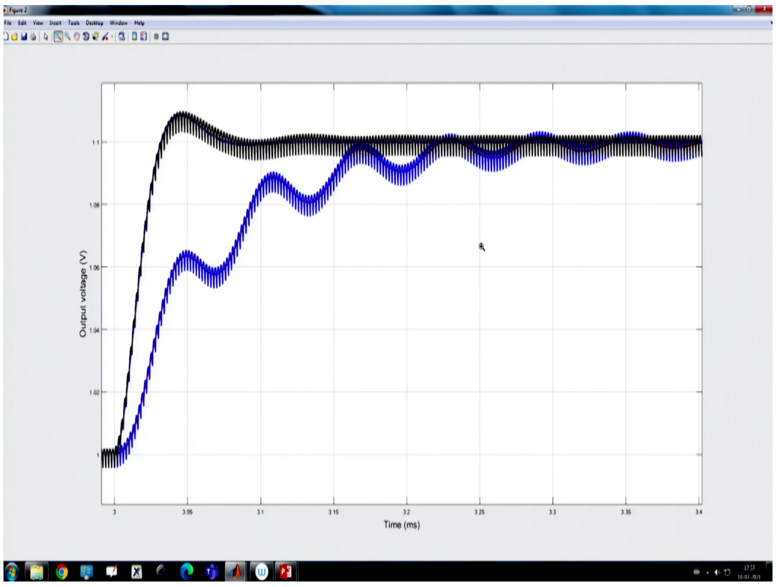


And, you can really well damped the system by using this.

(Refer Slide Time: 45:46)

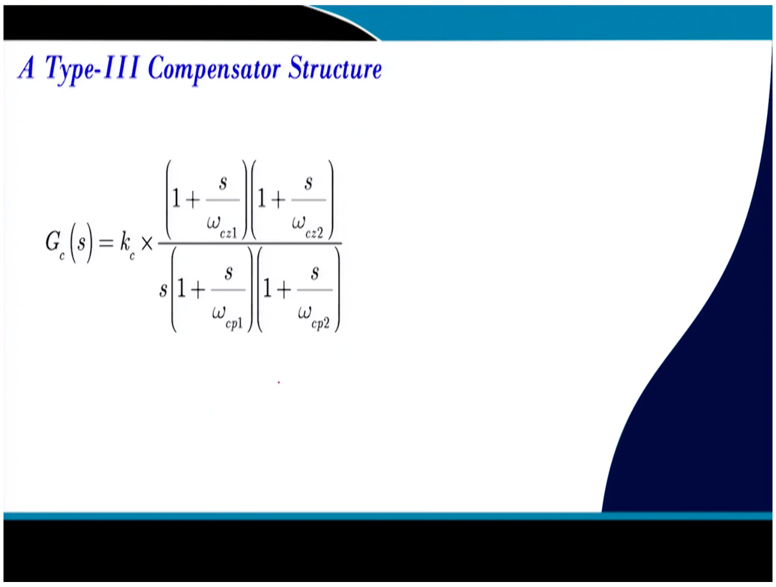


(Refer Slide Time: 45:49)



That means this is possible because we have an additional degree of freedom. So, the type III compensator gives an additional degree and PID control. We have a real difficulty for a wide compensating operating range.

(Refer Slide Time: 45:59)

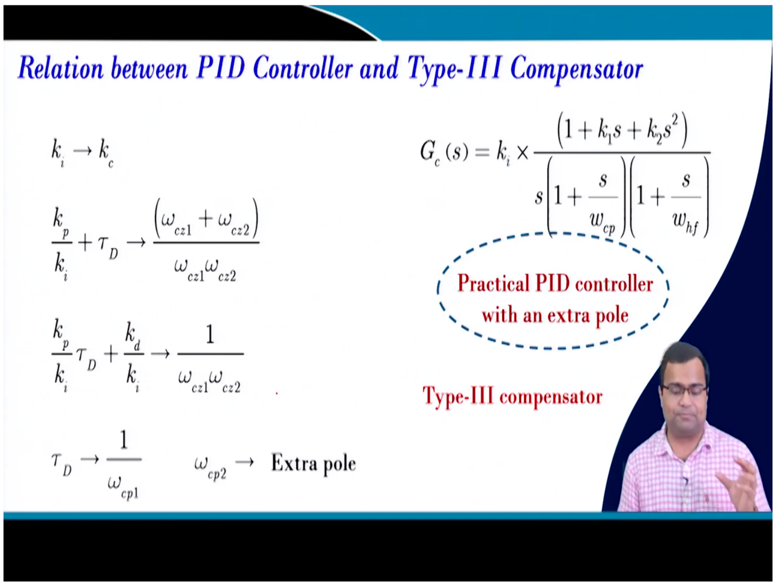


## A Type-III Compensator Structure

$$G_c(s) = k_c \times \frac{\left(1+\frac{s}{\omega_{cz1}}\right)\left(1+\frac{s}{\omega_{cz2}}\right)}{s\left(1+\frac{s}{\omega_{cp1}}\right)\left(1+\frac{s}{\omega_{cp2}}\right)}$$

So; that means, type III compensator will be used.

(Refer Slide Time: 46:00)



## Relation between PID Controller and Type-III Compensator

$k_i \to k_c$

$\frac{k_p}{k_i} + \tau_D \to \frac{(\omega_{cz1} + \omega_{cz2})}{\omega_{cz1}\omega_{cz2}}$

$\frac{k_p}{k_i}\tau_D + \frac{k_d}{k_i} \to \frac{1}{\omega_{cz1}\omega_{cz2}}$

$\tau_D \to \frac{1}{\omega_{cp1}}$ $\omega_{cp2} \to$ Extra pole

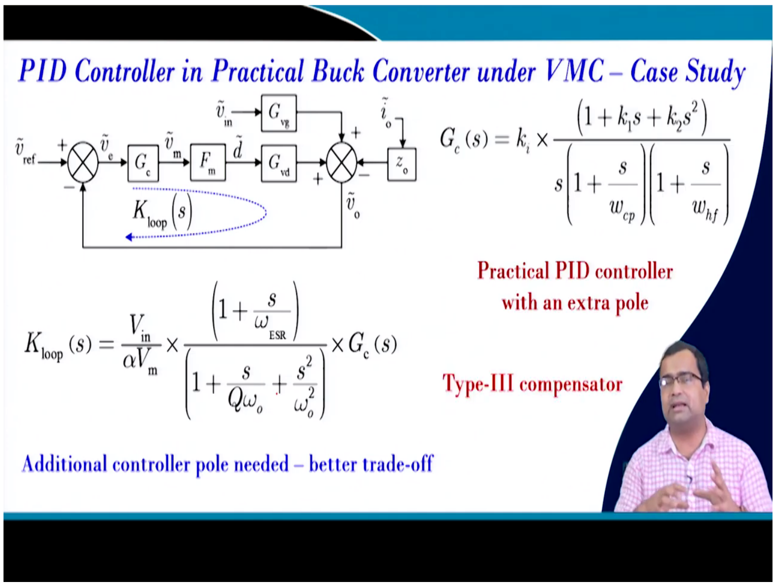
$$G_c(s) = k_i \times \frac{(1 + k_1 s + k_2 s^2)}{s\left(1 + \frac{s}{w_{cp}}\right)\left(1 + \frac{s}{w_{hf}}\right)}$$

Practical PID controller with an extra pole

Type-III compensator

And, which is nothing but a PID practical PID controller with an extra pole, ok.

(Refer Slide Time: 46:06)



## PID Controller in Practical Buck Converter under VMC – Case Study

$$G_c(s) = k_i \times \frac{(1 + k_1 s + k_2 s^2)}{s\left(1 + \frac{s}{w_{cp}}\right)\left(1 + \frac{s}{w_{hf}}\right)}$$

Practical PID controller with an extra pole

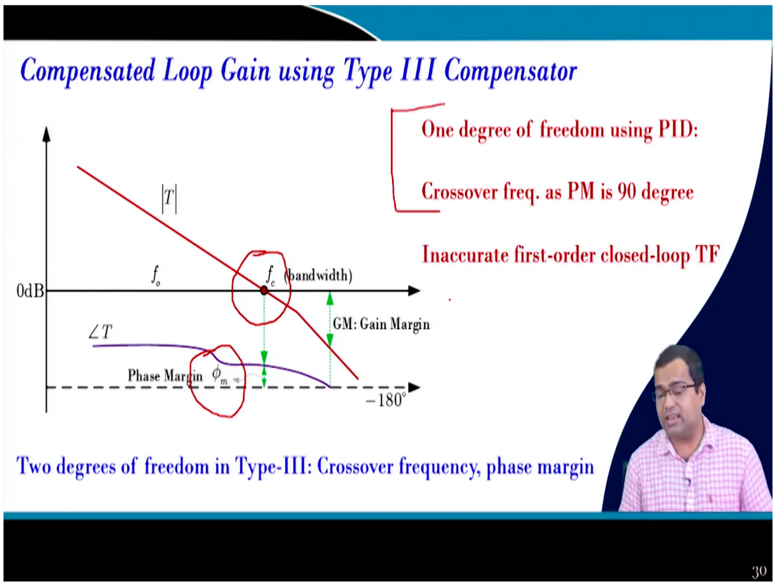
$$K_{loop}(s) = \frac{V_{in}}{\alpha V_m} \times \frac{\left(1 + \frac{s}{\omega_{ESR}}\right)}{\left(1 + \frac{s}{Q\omega_o} + \frac{s^2}{\omega_o^2}\right)} \times G_c(s)$$

Type-III compensator

Additional controller pole needed – better trade-off

Now, so, I have shown you the case study the better trade-off can be obtained by a compensator and we will detail discuss the design in the next class.

(Refer Slide Time: 46:15)


## Compensated Loop Gain using Type III Compensator

One degree of freedom using PID:

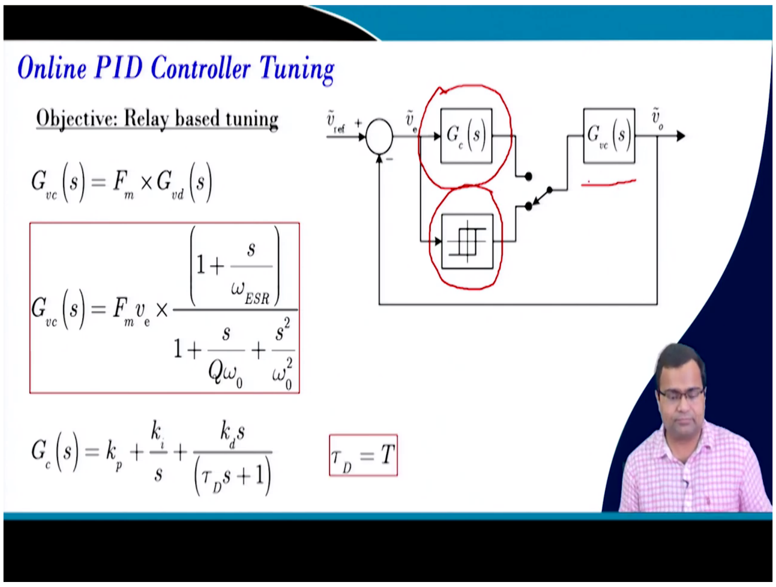
Crossover freq. as PM is 90 degree

Inaccurate first-order closed-loop TF

Two degrees of freedom in Type-III: Crossover frequency, phase margin

Now, here the additional degree of freedom can be used to independently adjust both the cutoff frequency as well as the phase margin using these two. So, earlier it was only 1 degree of freedom, where we have only control over the cutoff frequency, but it has a limit, but now so, inaccurate first-order model in analytical method.

(Refer Slide Time: 46:42)


## Online PID Controller Tuning

Objective: Relay based tuning

$$G_{vc}(s) = F_m \times G_{vd}(s)$$

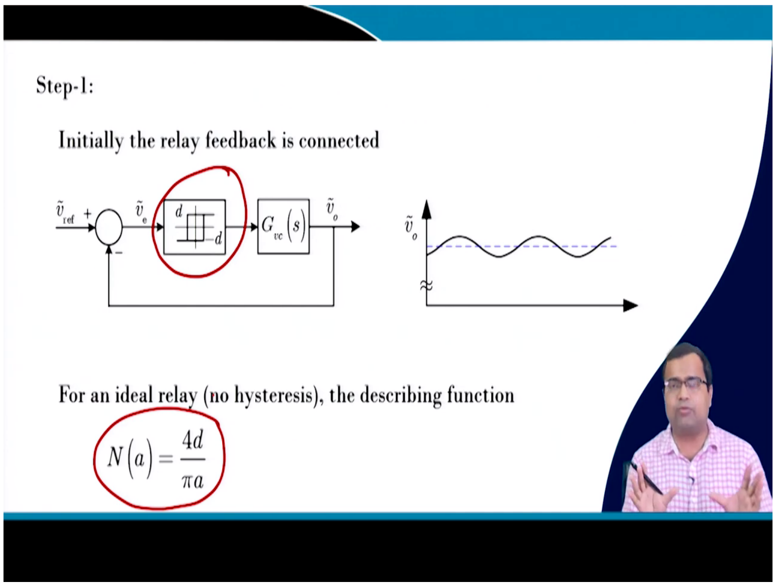
$$\boxed{G_{vc}(s) = F_m v_e \times \frac{\left(1+\frac{s}{\omega_{ESR}}\right)}{1+\frac{s}{Q\omega_0}+\frac{s^2}{\omega_0^2}}}$$

$$G_c(s) = k_p + \frac{k_i}{s} + \frac{k_d s}{(\tau_D s+1)} \qquad \boxed{\tau_D = T}$$

But, now with this type III compensator, we have an additional degree of freedom and we could achieve a much better response. Online controller tuning is relay based tuning that we discuss. This technique can be used in real time. In fact, it has been used in many research paper probably in commercial product, where the PID controller can be tuned in real time by introducing a hysteresis nonlinearity. Even if you make the hysteresis band very small, it becomes a relay nonlinearity.

And, that can be enabled before you actually connect the actual. So, this is your actual PID controller. But initially relay is connected, and it creates a sustained oscillation and your this is your G vc which includes your G vd into modulator gain, ok.

(Refer Slide Time: 47:34)



Step-1:

Initially the relay feedback is connected

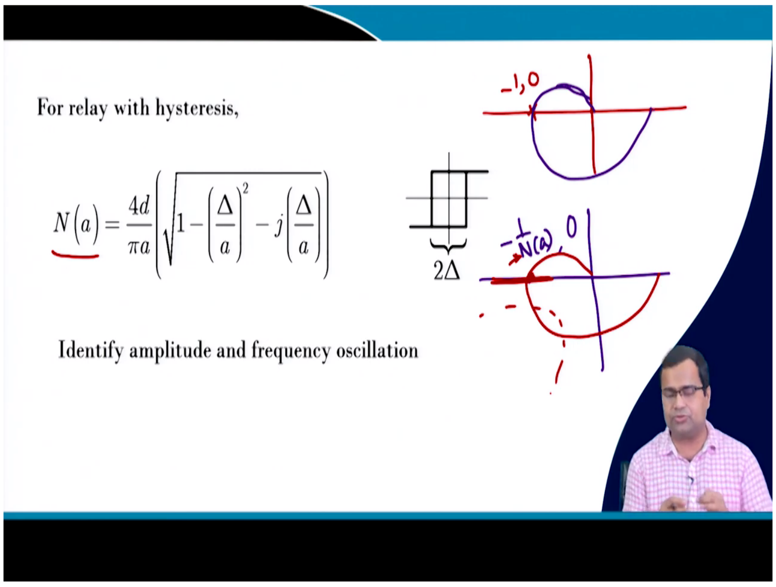
For an ideal relay (no hysteresis), the describing function

$$N(a) = \frac{4d}{\pi a}$$

So, step 1 initially relay feedback is connected and due to this relay action, but you need to make sure that your relay, you know, your DC gain should be sufficiently high. So, that you can actually take into the inside the relay band, then if you start this; the input amplitude of the relay should be sufficiently large. And, then you know this technique the analysis can be carried out using describing function method, by using a generalized Nyquist stability criteria and that we are not going to discuss here ok.

So; that means, using describing function, we can get the gain of this particular block we can obtain the gain. This is for a pure you know relay where there is no hysteresis.

(Refer Slide Time: 48:25)



For relay with hysteresis,

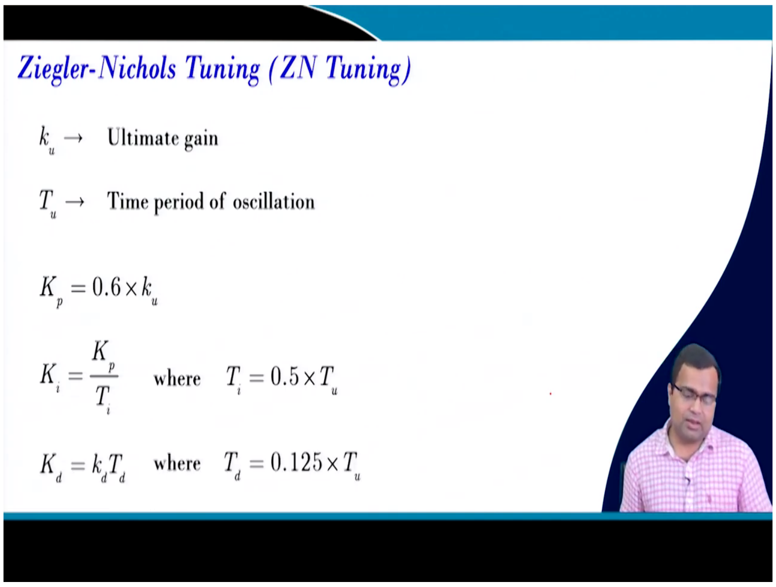
$$N(a) = \frac{4d}{\pi a}\left(\sqrt{1-\left(\frac{\Delta}{a}\right)^2} - j\left(\frac{\Delta}{a}\right)\right)$$

Identify amplitude and frequency oscillation

-1, 0
$2\Delta$
$-\frac{1}{N(a)}$, 0

But with hysteresis will get a complex term.

(Refer Slide Time: 48:30)



## Ziegler-Nichols Tuning (ZN Tuning)

$k_u \rightarrow$ Ultimate gain

$T_u \rightarrow$ Time period of oscillation

$K_p = 0.6 \times k_u$

$K_i = \frac{K_p}{T_i}$ where $T_i = 0.5 \times T_u$

$K_d = k_d T_d$ where $T_d = 0.125 \times T_u$

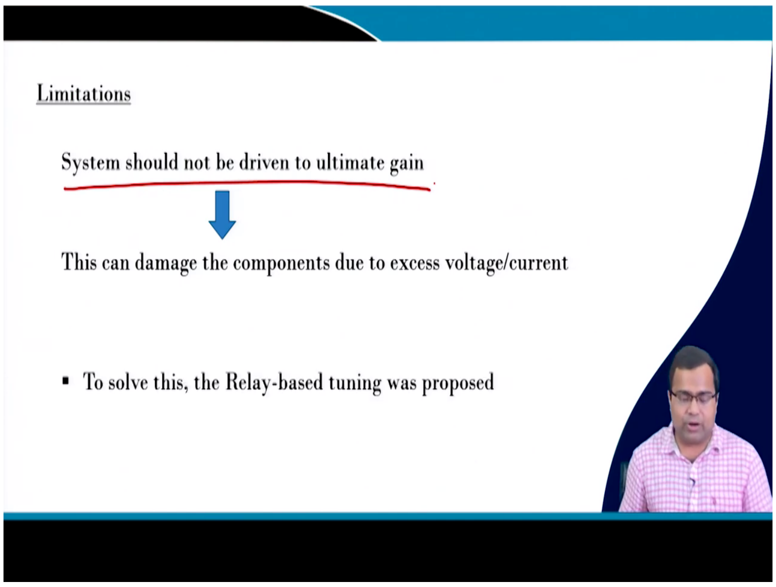
And, then by plotting the Nyquist plot of the linear plane and also the Nyquist plot for this the gain of this relay hysteresis term. We can find out the intersection point and from that intersection point by finding out, you know, what is the phase and amplitude. We can find out the amplitude of oscillation of the, but we have to make sure that it should satisfy the extended Nyquist stability criteria; that means, you know we know that the Nyquist stability criteria there is a minus 1 0 point.

And, if we talk about any linear transfer function and if it intersects like this, if it intersects like this, then this is a critical point. So, this critical point; that means, if it does not encircle and if there is no unstable pole, then system will be stable, but if it encircles if the open loop does not have any unstable pole, then it will be unstable. But, in case of our relay feedback along with this, you know this minus 1 it got changed, it instead of minus 1 it is minus 1 by n a comma 0.

And, then we consider the linear plot the Nyquist plot. And, the requirement is the intersection. If it intersects, then only there will be a possibility of limit cycle oscillation. This minus N by a it can be varied by varying amplitude and in this case also frequency dependent term it will be something like this.

So, we can actually find out whether there exist a limit cycle and if it exists, what is the amplitude and oscillation. Then, if we find out a control oscillation, we can find out ultimate gain, ultimate time period and we can find out k p k i k d using Ziegler Nichol Tuning Method.

(Refer Slide Time: 50:31)



Limitations

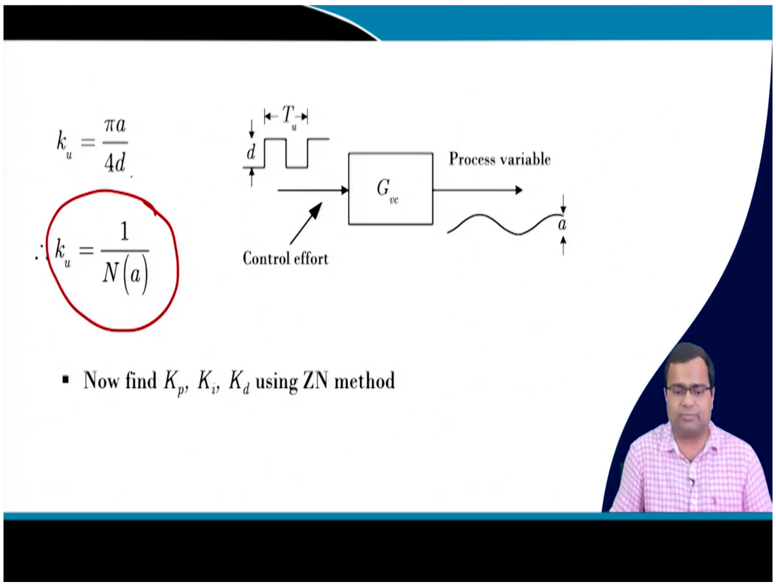
System should not be driven to ultimate gain

This can damage the components due to excess voltage/current

- To solve this, the Relay-based tuning was proposed

So, this can be used in real time, but limited the system should not be driven in ultimate gain in a standard technique, like Ziegler Nichol tuning rule. So, this can damage the component in order to solve the relay based tuning was proposed.

(Refer Slide Time: 50:45)


$$k_u = \frac{\pi a}{4d}$$

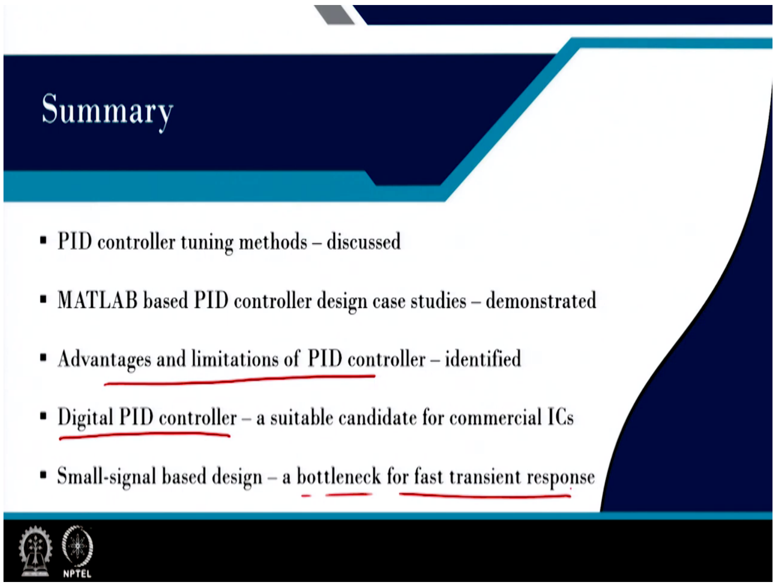
$$\therefore k_u = \frac{1}{N(a)}$$

$T_u$
$d$
$G_{vc}$
Process variable
Control effort
$a$

- Now find $K_p$, $K_i$, $K_d$ using ZN method

And, this can be used to generate the control oscillation and then critical gain can be formulated by 1 by this 1 divided by this gain. And, then we can use the earlier Ziegler Nichol tuning method to find out the individual parameters of the K p K i K d.

(Refer Slide Time: 51:05)


## Summary

- PID controller tuning methods – discussed
- MATLAB based PID controller design case studies – demonstrated
- Advantages and limitations of PID controller – identified
- Digital PID controller – a suitable candidate for commercial ICs
- Small-signal based design – a bottleneck for fast transient response

So, with this I want to summarize that PID controller tuning methods are discussed, MATLAB based PID controller design case studies were demonstrated, advantage and limitations of PID controllers are identified, digital controller can solve many of these techniques. Because we have considered a type III compensator, in digital control we do not require. Even a digital PID controller will have because it is sufficient to anticipate and the parameter can be adjusted in real time.

So, that is a suitable candidate for commercial IC. And, the small signal based design has a bottleneck for faster transient response. So, which also you know in future subsequent lecture we will see the large signal based PID controller tuning, where the PID controller can achieve.

Because we are still talking about bandwidth in terms of up to 1 10 switching frequency or even less, but we can go up to the critical limit by setting the PID controller using non-linear tuning that will also discuss. So, with this I want to finish it here.

Thank you very much.